AD-A207 875

ORT DOCUMENTATION PAGE

1a. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR-89-0617
6a. NAME OF PERFORMING ORGANIZATION: Univ of Massachusetts	6b. OFFICE SYMBOL (If applicable):
6c. ADDRESS (City, State and ZIP Code): Computer & Information Sciences Dept. Amherst, MA 01003	7a. NAME OF MONITORING ORGANIZATION: Air Force Office of Scientic Research
	7b. ADDRESS (City, State and ZIP Code): Building 410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR	8b. OFFICE SYMBOL (If applicable): NM
	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-86-0021
8c. ADDRESS (City, State and ZIP Code): Building 410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A7	

11. TITLE (Include Security Classification): Recognizing 3D Objects from 2D Images Using Structured Knowledge Base of Geometric Views

12. PERSONAL AUTHOR(S): Allen R. Hanson

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
FINAL	FROM 10 Oct 85 TO 31 Aug 88		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Model-based object recognition is an essential task for mobile robotics and assembly. Given an image of a scene containing one or more objects from unknown viewpoints, the goal is to efficiently recognize those objects for which there is sufficient evidence. At the University of Massachusetts, we are developing a model-based object recognition system which is capable of recognizing objects from a large data base of models and from arbitrary viewpoints.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Dr. Abraham Waksman	(202) 767-5027	NM

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE. UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Final Technical Report

for the period 10/1/85 - 8/31/88

Grant Number: AFOSR-86-0021

submitted by

Principal Investigator: Allen R. Hanson
Co-Principal Investigator: Edward M. Riseman

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

1. Overview

Model-based object recognition is an essential task for mobile robotics and assembly. Given an image of a scene containing one or more objects from unknown viewpoints, the goal is to efficiently recognize those objects for which there is sufficient evidence. At the University of Massachusetts, we are developing a model-based object recognition system which is capable of recognizing objects from a large data base of models and from arbitrary viewpoints.

The first part of the recognition problem is extracting useful data from an image. If one has a scene with controlled lighting or only a few types of objects, then this is easily solved [LOW85,HUT87]. However, in the domain of natural scenes, this is a much more difficult problem and requires more sophisticated algorithms even to extract straight lines. In Section 2. of this report, we present the results of an algorithm for extracting straight lines. This algorithm demonstrates how the principles of perceptual organization can be used to reduce the computation involved in this process.

The second part of the recognition problem is the acquisition and representation of three-dimensional models. Only a small part of that problem is dealt with here. In Section 3. we present a mathematical formulation of the problem of reconstructing a curved surface from multiple views with known camera motion. An important issue in the representation of objects is the relationship between different views of the same object. In Section 4. we present a discussion of the construction of the viewing sphere of generic views, which is the dual of the aspect graph [KOE76,GIG88,BOW88], for smooth surfaces. In future work, we hope to combine the two ideas to produce a view sphere representation of an object directly from a sequence of multiple views.

The third part of the recognition problem is reducing the complexity of the matching process. For many applications, the number of objects could range from 100 - 1000, and the number of distinct views for each object could range from 10 - 100; thus, we could be faced with 1K - 100K different object views. This requires that the recognition process and the representation be very efficient. This type of efficiency in processing can be achieved by using the image features or tokens as an index into the model base to retrieve the most likely candidates. In Section 5., an analysis of generic views or aspect graphs is used to construct a prediction hierarchy which provides this index. Hypothesis generation using the index is followed by a verification process in which the position and orientation of each object is computed more precisely, and a more complete matching between model and data is obtained.

2. Extraction of straight lines

This section presents a computational approach to the extraction of straight lines based on the principles of perceptual organization. In particular we consider how local information that is spatially distributed can be organized into large scale geometric structure in a computationally efficient manner. Symbolic tokens representing line segments, and relations which are primarily geometric in nature, are used to control a hierarchical grouping process. The relational measures on pairs of lines are based on collinearity, proximity, and similarity in contrast.

The algorithm is implemented within a local, parallel, hierarchical framework for symbolic grouping that involves a cycle of linking, optimization, and replacement steps. The initial line segments are generated via gradients at the position of zero-crossing contours of a Laplacian operator. The local linking process forms a line graph of tokens by applying a set of relational constraints, to the line tokens in a neighborhood around each line token. By controlling the constraints, the number of lines that must be examined as possible extensions of the given line can be filtered down to a much smaller number. The optimization process uses a least-squares error measure to select the best straight line fit through all sequences in a link graph of the remaining candidates. If the best line fit has sufficiently low error, the participating tokens are used to form a new token in the replacement step.

As the grouping cycle proceeds, shorter line tokens are aggregated into longer line tokens, creating a hierarchy of levels based on scale. Selection of appropriate values for parameters in the grouping process are examined with respect to computational efficiency and empirical analysis. Experimental results on a variety of natural scene images demonstrate effectiveness of the filtering and optimization stages in the extraction of straight lines. Issues in the development of a more general framework for symbolic grouping are also discussed.

2.1 Introduction

It is clear that not all processing is or should be *directly* related to three-dimensional interpretation. If early stages of two-dimensional processing are capable of producing symbolic descriptors that are usable by higher level processes, a great deal of flexibility will be gained in dealing with difficult problems in vision. In addition, there are varying goals in visual perception. In navigation, one needs to know *where* objects are in space, not necessarily *what* they are. However, for recognition, the case may be reversed; the viewer usually does not need to know how far away each part of an object is to recognize it. Moreover, the human visual system has no difficult in dealing with both tasks simultaneously. This suggests that it should be possible to use symbolic grouping mechanisms for two-dimensional processing without the necessity of first constructing a three-dimensional representation of the data.

This study seeks to bring together themes in intermediate level vision for the effective extraction of a straight line representation of an image. While the principles of percep-

tual organization were described by Wertheimer[WER23] and have been discussed with respect to computer vision[LOW85,MAR82,ROS86], their importance has not been widely recognized and employed. In particular, the ideas have not been fully translated into a computational model. Our goal is to transform the Gestalt laws of perception into a computational framework which supports the symbolic grouping of line tokens in a robust and efficient manner.

An alternative view of the research presented here is that of an optimization process for fitting long straight lines to subsets of short lines. Since the set of lines in an image is very large – initially equal to the number of candidate short edges at the beginning of the process – the number of interesting candidate subsets that must be examined for global line fits is impossibly large. The role of the grouping mechanisms is to apply geometric constraints as filters and focus the optimization process on a small number of interesting candidates in a hierarchical manner, in order to make the approach computationally feasible. From this view, the evaluation of the grouping criteria should be based on the degree of efficiency and the error of fit in the lines that are found (i.e. how close the grouped lines are to the "best" straight lines that are extractable).

The algorithm that we develop in this paper will have the following processing characteristics: it will be a locally parallel algorithm operating on a hierarchy of geometric symbols, or tokens, that are associated with two dimensional image events. Groups of tokens at some level L in the hierarchy will be replaced by a single token at level $L+1$, where a level in the hierarchy corresponds to an iteration of the grouping process. The Gestalt laws suggest relational constraints which can be used as filters in the search for groups of line tokens that are likely candidates to form a longer straight line. These constraints rely heavily on geometric relationships among tokens and are based on the computation of relational measures, such as difference of orientation or percent of spatial overlap of pairs of tokens. Repeated grouping leads to the description of image tokens at multiple scales. Note that our use of the term "scale" differs from that of Witkin[WIT83], where scale was related to the size of the Gaussian function used to smooth an image. Here, we view scale as the size of the area in the image which is being represented by each token. These two notions of scale are independent; the former being related to the frequency spectrum, while the latter can be described as symbolic. Stevens and Brooks [STE87] have presented evidence from perceptual grouping of texture that symbolic grouping of discrete tokens plays a significant role in human visual perception.

The idea of grouping line segments is not new. Nevatia and Babu [NEV80] have reported criteria for grouping edges; however, they only applied it locally and they did not take into account more global context. Their system used thresholding and thinning to reduce the number of edge pixels, then adjacent edge pixels were linked based on orientation and location with respect to each edge pixel. Chains of a fixed length were approximated by straight lines.

In the SCERPO system, Lowe[LOW85,LOW83] has used grouping to find collinear and parallel lines for matching. He has tried to deal with the problem of how to determine which scales contain the information needed to solve a specific problem. For example, if

one has a description of some objects which might be in a scene, and one does not know *a priori* how large the objects are in the image, how can one determine at which scale to look for evidence for those objects which are present? In his papers, he analyzes the problem of segmenting a curve into straight and circular line segments. The solution he proposes is a measure of *meaningfulness* based on probability. Events are considered meaningful if they have a low probability of occurring when the scene does not contain the desired objects and a high probability when they do. In the case of points lying on a curve, the former probability is based on the assumption that edge points which do not lie on the same straight or curved line will be distributed uniformly in the image. Since this assumption often will be violated in some cases, it might be possible to find a weaker assumption which would still permit a mathematical formulation.

McKeown and Pane [MCK85] have also devised rules which are similar to ours and have applied them to the problem of linking elongated regions. They use the criteria that regions a) must be close; b) must not overlap too much; c) must have similar orientations, and d) must have small lateral distance. Thus, they implement the rules of collinearity and proximity. A major difference between their approach and ours, however, is that while they used the size of each region to determine its context, they do not use a hierarchy. In addition, there is strong basis for the conjecture that region tokens are not as reliably extracted with respect to the placement of their boundary, and therefore may not be as effective in the grouping process.

Rosenfeld [ROS86] has described how many of the principles of perceptual organization can be used by pyramid-based algorithms in order to achieve a hierarchical implementation. In particular, Hong, Shneier, Hartley, and Rosenfeld [HON83] describe such an implementation of a pyramid-based algorithm for grouping line segments. The single criterion used by their algorithm for grouping lines, which are in adjacent cells in any level of the pyramid, is that they have similar orientation. This differs from our algorithm in which a more detailed analysis of collinearity and proximity is used.

It should also be noted that the grouping algorithm developed here fits within many different larger frameworks for image interpretation. For example, the VISIONS system at the University of Massachusetts is a general system for knowledge-based scene interpretation[DRA89,HAN87,RIS86]. This leads to a three-level abstraction hierarchy as the underlying conceptual model for image interpretation: the low (pixel) level, intermediate (token) level, and high (object hypothesis/knowledge) level. The construction and grouping of tokens described here is part of the intermediate level of symbolic processing.

Section 2.2 of this paper discusses computational considerations in the development of a symbolic line grouping process. We examine issues related to the use of symbols, hierarchical processing, and efficiency. In Section 2.3 the algorithm for symbolic hierarchical grouping of line tokens is developed using relational constraints (primarily geometric) between line tokens. Section 2.4 presents experimental results demonstrating the effectiveness of the algorithm in extracting straight lines, including long low-contrast lines. In Section 2.5 we discuss how the computational framework can be generalized into symbolic grouping of more complex and abstract geometric structures.

2.2 Computational Issues for Symbolic Line Grouping

In this section we seek to motivate the general computational framework within which our line grouping algorithm has been implemented. In particular, we must consider how local information that is spatially distributed can be organized into large scale geometric structure in a computationally efficient manner.

There is a gap in representation and in spatial scale between pixels at the low-level and semantic contexts at the high level. A symbolic processing framework provides a geometric basis for the grouping of related tokens in a way that pixel-level processing cannot accommodate. By moving to a symbolic representation a single token can represent information from a large number of pixels. This begins to deal with the local-global spatial issues for relating information.

Consideration of the following issues at the low intermediate levels of visual processing are relevant:

1. integrating local information into consistent global structure,
2. data abstraction,
3. efficient performance,
4. multiscale geometric description of image events.

2.2.1 Integrating Local Information into Consistent Global Structures

Probably the key consideration is how to achieve consistent integration of unreliable or ambiguous information across different and possibly large parts of an image. The image events which provide evidence for objects are often not present in contiguous pixels, and the contextual information that is relevant can appear at an arbitrary scale. While local processing in a small neighborhood is necessary as the starting point, it is not sufficient. Figure 1a-d shows some examples of distributed image events that need to be integrated into global lines. In this figure each set of events are perceived as straight lines at some scales, but not at others. Figures 1a and 1b show lines that are fragmented, but which seem continuous when viewed from a distance. Figure 1c shows lines whose directions vary, but when viewed from a distance they are seen as straight. Figure 1d shows a sequence of dots that are perceived as a straight line. In fact the lines on this page are composed of dots. In each of these cases the evidence for a straight line is not present if processing is restricted to a small window; there are gaps or the orientations are not aligned. If we integrate the information present in a large number of these windows which are properly aligned with what we perceive as the line, then the evidence is very strong. The use of geometric context facilitates the process of integrating distributed information.

2.2.2 Data Abstraction

In low-level vision, we deal with enormous amounts of data. Obviously, the information has to be organized in such a way that the essence, or an overview, of the image structure is available, as well as a quick access to finer levels of detail. Among the ways of "summarizing" the data are *filtering* and *abstraction*. Filtering reduces the amount of data, but it may also throw away important information. In particular, filtering of line tokens via constraints on attributes [DRA89,HAN87,RIS86] only removes tokens, and it is often necessary to create new ones which are more likely to be significant.

Abstraction is more than data reduction; it involves organizing the information in a way that makes it more accessible for further processing. Unlike filtering, the process of abstraction involves *replacing* a group of tokens by a single token, or a complex object by a simpler one. For example, consider the following geometric description of a checkerboard: "the smallest meaningful units are squares; those squares lie on straight lines; the lines form a regular pattern covering a certain area." Note the increasing scale in size and abstraction for the descriptions, which are all valid simultaneously. They are symbolic, based on geometric relationships. Our goal in this paper is to create a hierarchy of symbolic structures, where each level in the hierarchy has less data than the next lower level. The process of data abstraction can be viewed as creating more abstract descriptions of larger scale, rather than the process of reducing the spatial resolution of an image.

The control problem for the abstraction processes is still a difficult one. In general, there may be many possible ways to represent the information present in an aggregate of tokens; i.e. how does one choose the 'best' abstraction? Leeuwenberg has formulated a measurement of conciseness for representations of structural information; this seems to be related to the ease with which humans perceive groups of lines as an object [LEE78]. Such measures may be of use in evaluating specific abstractions.

In our domain of straight lines grouping, the representation will remain constant, i.e. a straight line token will be used throughout the abstraction process. We will employ an optimization procedure to choose those aggregations, among competing local candidates for line grouping, which are best via a least-squares error criterion.

2.2.3 Efficient performance

When one considers the huge amounts of sensory data which must be processed and the large number of potentially significant image events which need to be detected for typical visual tasks such as navigation and object recognition, issues of computational tractability must be considered. The process of abstraction helps because the data transformation involves grouping events which can be described more concisely as a unit, so that there are fewer items to process. However, in addition to reducing the amount of data, the abstraction processes themselves should be efficient, although not at the expense of generality.

Generalized Hough transform techniques are one way of performing rapid search and are capable of finding certain parameterized structures (e.g. lines, rectangles or circles)[BAL82,BAL83,D

However, the Hough transform is not a general solution to the problem of line token extraction. It ignores the spatial and geometric context in which the primitive is embedded, like the distance between tokens, or the location of other tokens that lie between the tokens to be grouped. To include all the significant geometric parameters in a generalized Hough space would increase the dimensionality beyond a feasible level.

This leads us to the question, "What is the necessary scope in the image needed to detect 'the whole'?" Typically, it should not be necessary to examine the entire image. On the other hand, the local view often cannot be restricted to just a pair of tokens (e.g. a token and a nearby neighbor). Depending on the type of grouping, the amount of context used will depend on the complexity of the new token to be formed and the computational resources required to examine all of the combinations within this contextual area.

What does it cost to find a new geometric structure involving a symbolic token A? The set of tokens around A which are examined in this search is called a *neighborhood*. Let us assume we want to detect the best combinations of A with its neighbors which satisfy specific criteria. The amount of computation for blind search would increase very rapidly with the number of tokens: if we searched in areas with n neighbors around every token and looked for combinations of m tokens, then the amount of computation would be of the order $C(n, m)$, which grows as n^m. How can the search space be reduced? One option is to reduce m, the number of tokens which compose a new structure. This can be done by building structures hierarchically. Another option is to reduce n, the number of tokens examined. This can be done by locally filtering the set of tokens to be searched based on token features, or searching only in appropriate directions or areas. These constraints can reduce the number of combinations in the neighborhood. In addition, it is possible to process neighborhoods in parallel, considerably reducing the time.

2.2.4 Multiscale geometric description of tokens

The hierarchical line grouping algorithm that we will present in the next section deals with the issues of local-global construction, data abstraction, and computational efficiency. All of these issues are related to hierarchical computation which will be achieved in our formulation by organizing the processing of line tokens along geometric scales.

The term *scale* refers here to a range of sizes of symbolic tokens with respect to the entire image. Computation involving a particular token will be restricted to a neighborhood whose scale can depend on the goals of the process and the tokens already found. This allows the processing of tokens of different sizes to have different parameters in the grouping process. However, it also introduces the need for combining the information at different scales in the hierarchy. For example, the final description of our checkerboard example discussed earlier encompasses information at many scales. (Note that the concept of size scale, as used here, is somewhat independent of the concept of abstraction scale; the description of the checkerboard has both multiple size scales and multiple abstraction scales.) The integration of distributed information in the image is accomplished in our system by performing processing sequentially at multiple scales, and restricting the pro-

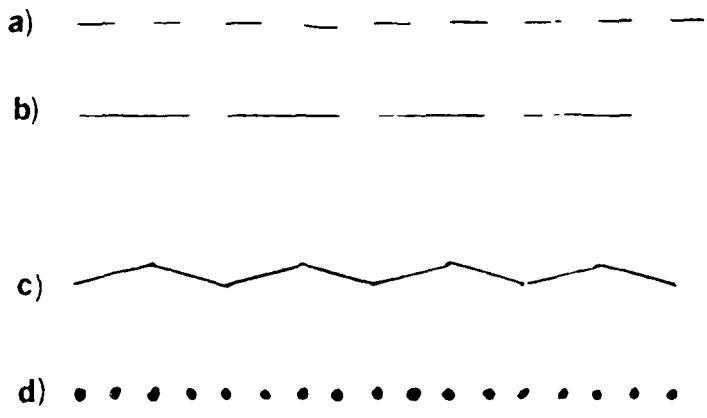


Figure 1: Perceptual Integration of Distributed Image Events. Each part is a group of events which can be perceived as a straight line at a sufficiently large scale.

cessing at each scale to the output of processing at lower scales, (although the processing at any scale can be spatially parallel). The reason for this ordering of scales is that our approach is both hierarchical and constructive. Note that this is in contrast to Fischler and Bolles [FIS83], who perform an analysis using multiple scales on curves which have already been produced, so each scale can be processed independently. Reynolds and Beveridge [REY87] also process all scales simultaneously by defining the neighborhood of a line to be dependent on the length of the line and refer to this relational measure as spatial proximity.

2.3 Applying Token-Based Abstraction to Straight Line Grouping

In this section we apply the principles of intermediate-level symbolic processing just presented to the problem of grouping lines into longer, straight lines. The need for a multiscale description suggests a hierarchy, which when combined with a symbolic representation, facilitates the satisfaction of the other processing requirements. For hierarchical, symbolic processing, neighborhoods at each level have a small number of image events, thus the search space can be kept small. At the same time the neighborhoods at higher levels cover a large part of the image, so integration of distributed events is also achieved. Having explained why symbolic processing or abstraction would be useful for an intermediate-level vision system, we now describe such a system. It consists of extracted image events called *tokens*, and operations on tokens called *abstractions* which produce new tokens.

This is demonstrated on the images of natural scenes (see Figure 6) for which standard line extraction techniques encounter significant problems due to noise or texture, or where edge events have low contrast. Our goal is to show that the significant straight lines in an image can be found using a geometric grouping algorithm via token aggregation and replacement [WEIS86,WEIS85]. The results of this straight line grouping algorithm have been used for the interpretation of natural scenes [DRA89].

2.3.1 Image tokens and the Grouping Process

A token is a description of a set of image events or an aggregation of other tokens. The goal is to create line tokens corresponding to those events which can be perceived as a unit. There are two ways to produce line tokens. They can be generated directly from an image, e.g. by edge operators, or they can be generated from other line tokens, e.g. straight lines can be grouped into longer straight lines. Since tokens which have a large size may require several steps to construct, it will be necessary to have tokens of intermediate size as precursors.

The major component of the algorithm is the hierarchical grouping process which has three steps that are performed at each level: *linking*, *optimization*, and *replacement*. The linking stage corresponds to the generation of hypotheses which satisfy a conjunction of binary relations and the verification of the line hypothesis using a measure of straightness, respectively. The binary relations are derived from *relational measures* [RIS87], which are general functions of the attributes of a set of tokens, based upon collinearity, proximity, and similarity in contrast; the specific relational measures used are described in more detail in Section ? There are many different decision functions which can be used to combine the relational measures into relations. These binary relations are used only to filter tokens while forming a line graph of locally related token pairs, and thereby focus in the search process when replacing aggregates of tokens by new tokens. Thus, the only requirement is that these filters reduce the search space but do not eliminate any likely candidates for replacement.

Straightness is based on a least-squares optimization criterion, and will be applied to line sequences of varying length, not just to pairs of tokens. Thus, we take the following algorithmic definition of a straight line: it is composed of a sequence of line segments in which consecutive pairs satisfy the relations of collinearity, proximity, and similarity of contrast; the entire sequence has the least error locally among the candidate groupings; and the error for the best sequence is acceptably low (i.e. passes a straightness test). All of these criteria may depend on scale; long lines, for example, can be separated by a larger gap than small ones and still be considered close.

For estimating the straight line that best fits the data, the line which minimizes the mean square error is used. This is not the only possible candidate. Kamgar-Parsi and Kamgar-Parsi[KAM87] and I. Weiss[WEI86] have found algorithms for fitting straight lines to clusters of data points in situations where there are a significant number of outliers or the distribution of distances from the true line is not Gaussian. While the noise in the image intensity values is not Gaussian, the errors in edge orientation seem to be approximated reasonably well by such a distribution. It should be noted that in our method, because the lines are filtered before straight line estimation is done, the likelihood of outliers is very low. In particular, since the mean square distance is especially sensitive to outliers it is a good metric for eliminating them, which is what is done.

2.3.2 Edge Detection

The input to the grouping algorithm is a set of line tokens together with a measurement of the average intensity contrast from one side of the line to the other. Usually this is produced by an edge detection algorithm applied to the image. The properties which an edge detection process should satisfy are: 1) high positional accuracy of an edge, even with aliasing, 2) good sensitivity to high frequency data, and 3) reduction of the data (many pixels don't produce edge points). The last of these properties, reduction of data, can make a significant difference in the computation time. However, it is also important that the algorithm be sensitive to low-contrast edges, so that the usual techniques of thresholding edges of low gradient magnitude is not acceptable except at values that are clearly down at the level of noise.

There are many possible edge detection algorithms which could be used. The two algorithms which we have used for selecting initial edge locations are zero crossings of the Laplacian operator and a directional edge operator based on the work of Haralick and Canny [HAR84,CAN86]. Although there may be algorithms which have better performance in some cases, these two are representative of a class of algorithms; most of them are expected to have the same types of problems which we encounter here [DAV75]. The Laplacian operator was chosen because in early experiments it seemed to be the better of the two for our purpose, although no quantitative evaluation was performed.

First, the image is convolved with a 3×3 mask which approximates the Laplacian:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Next, zero crossings of the Laplacian image are detected and their positions determined by linear interpolation. A zero crossing is a zero value of the Laplacian output, and consequently is the location of a change in sign; it typically occurs within a pixel and consequently requires subpixel interpolation in order to determine its spatial position. Figure 7a shows that the zero crossing contours do not consistently follow the boundaries of objects or other identifiable parts of the image (e.g. see the house roof). The reasons why the contours are not necessarily a reliable indicator of the direction of an edge can be understood by looking at some of the assumptions about edges and the Laplacian operator. In Marr's paper [MAR82] on the use of the Laplacian, he makes the assumption that the intensity surface is planar in a neighborhood of the edge. Even for an ideal curved intensity step edge this assumption will be violated, and the results in real images with aliasing from the digitization process have serious problems. On the other hand, the gradient tends to remain perpendicular to the edge. For this reason, we use the gradient direction at each edge point for the edge orientation; in general, this will differ from the direction of the zero-crossing contour. Figure 7b shows the gradient information for the edges in the house roof scene.

The location of a zero-crossing point is computed by interpolation of the values at four pixels which form the corners of a square. The gradient at that point is also computed by interpolation. An edge is positioned with its center at the zero-crossing point, and the orientation of the edge is perpendicular to the gradient. Each edge can now be viewed as a line token with a contrast (the gradient magnitude), direction, a start point, an end point, and is defined to have a length of one pixel.

The grouping algorithm that will be presented does not require that the input edges be completely reliable, and it has produced very good results despite the variety of problems encountered with the Laplacian operator. These problems include saddle points of the Laplacian and anti-edges at local minima. When a zero-crossing point is a saddle point of the Laplacian surface, then locally the contour will consist of a pair of hyperbolas or a pair of lines which cross. If these saddle points have low density, then they can be ignored. If they have high density, then a different type of grouping needs to be applied because the orientation at a saddle point is ambiguous (we do not consider that here.) Some experiments were performed in which saddle points were considered as edges with indeterminate orientation, and there was little effect on the results. Zero crossings of the Laplacian also occur where the gradient magnitude has a local minimum. The current implementation does not test for these *anti-edges*, and therefore they remain present in the initial edge data.

The process of acquiring an image itself can introduce artifacts which complicate the line grouping process. Subsampling of an image, quantization effects, and the presence of

noise can result in isolated pixels which have a high contrast with respect to their neighbors. Subsampling is especially noticeable when the pixel size of the sensor is large compared with the spatial period corresponding to the highest frequency of the intensity variations on the image plane. As a result of the local extrema produced by these phenomena, zero-crossing contours surround isolated pixels and produce a point-like structure. The line grouping algorithm presented here is not specifically designed to handle this situation, which would require the grouping of isolated points. In addition, noise can introduce numerous local maxima in the gradient of the intensity, producing multiple zero crossings, and hence multiple edges parallel to the visually significant one. Many of these problems could have been solved by smoothing with different-width Gaussian masks, which has been explored by Witkin and others [WIT83,YUI83,MAR82] using a scale-space approach. However, as mentioned earlier, smoothing removes details which are essential for some structures. For example, a very thin linear structure, even if it is long, will become undetectable if high-frequency data are filtered out. Our approach, on the other hand, is to use the geometric context to eliminate edges (due to subsampling, noise, anti-edges, etc) which do not form lines in the image. However, we do present some experimental results using Gaussian smoothing to provide some insight into these issues.

2.3.3 The Hierarchical Grouping Process

Our line grouping process uses geometric and nongeometric properties at multiple scales to form long, straight lines from shorter segments. In general, a grouping cycle consists of three steps: linking, optimization, and replacement. In the linking step pairs of line segments are connected in a graph data structure based on binary relations. Here, lines that are not likely to participate in straight line fits are filtered. In the optimization step, sequences of linked line segments are tested for straightness, and then a single line segment may be substituted for the sequence in the replacement step.

The issue is how to avoid searching and testing all combinations of line segments within a large area. The first part of the solution is the construction of the *link graph*. If there are N lines in the area to be searched referred to as the *perceptual radius* then there are 2^N sets of lines which could combine to form a straight line. On the other hand, if the density of lines, d, is uniform, and r is the perceptual radius, then $N = d\pi r^2$. If l is the average line length and L is the maximum number of lines in a sequence of linked lines inside the perceptual window, then $Ll \leq 2R$. Thus,

$$L \leq \frac{2\sqrt{N/\pi d}}{l}$$

so $L = O(\sqrt{N})$. If the average branching of the link graph is G, then the complexity of the search is only G^L, where G is usually less than 2.

The second way to reduce the computational complexity is to limit the number of tokens in the area being searched. This is done by processing the lines hierarchically: lines

are repeatedly grouped into longer lines, which implicitly defines a scale-space hierarchy. The search always remains local with respect to the current scale of processing, and there are fewer long lines over larger areas of the image.

Geometric Constraints for Linking

Linking is the first step in the grouping process and is the means by which we can reduce the search space by filtering the number of tokens to be examined in the subsequent processes. The linking step consists of a search for pairs of lines that satisfy the geometric and nongeometric relations which make them candidates for grouping. The criteria are based on relational measures: difference in orientation, difference in contrast, relative overlap, lateral distance, and distance between endpoints. Each of these is encoded in a heuristic function. The relational measures provide a means of comparing pairs of tokens via a continuous variable. Threshold parameters are used to convert each of these measures into binary relations. A link in a graph is defined from one token to the next if the pair satisfies all of the binary relations. Note that the thresholds defining these relations can be set loosely to avoid removing lines that might participate in the optimum straight line fit through the given line. Thus, while appropriate setting of these parameters is an issue, they can be set loosely with the goal of making the optimization step more efficient but not removing the best line fit. We will discuss the parameter settings later.

The geometric relationships that have been used are collinearity and proximity, and the nongeometric relationship is *similarity of contrast*. Consider a line $L_1 = (P_1, P_2)$. A Line $L_2 = (Q_1, Q_2)$ is linked to L_1 if it fulfills the following five relational constraints:

1. **Linking Radius.** L_2 must intersect the circular area whose radius is defined as the *linking radius*, R_l, and whose center is P_2 on L_1 (See Figure 2a). This radius should depend on the lengths of the lines and the density of line tokens. For computational efficiency, this test is an initial coarse filter to select nearby line candidates and is applied during the process of retrieving lines from the data base. Thus, the other tests are only applied to a small fraction of the lines. A more restrictive test for proximity is performed in step 3.

2. **Collinearity.** The lines must be approximately collinear. This relation has components for similarity of orientation and lateral distance between lines. a) The difference in the **orientations** of the two lines must be less than a threshold (See Figure 2b). In particular, lines which are collinear, but have opposite contrast, are 180 degrees apart and are not linked. This condition can be relaxed to allow such lines to be linked. b) The **lateral distance** between the two lines must be less than a threshold. The lateral distance is measured as the perpendicular distance from the midpoint of the line L_2 to L_1 (See Figure 2c).

3. **Endpoints.** The endpoints must be close. The projection of Q_1 onto L_1 must lie within a specified interval, which is proportional to the length of L_1. (See Figure 2d).

4. **Overlap.** The lines must not overlap too much. Let P_1 and P_2 define an order on L_1, P_1 being *before* P_2. If Q_1 projects onto L_1 between P_1 and P_2, then the distance from the projection of Q_1 to P_2 is the overlap, and must be less than a fixed percentage of the length of L_1 (See Figure 2e).

5. **Contrast.** The contrast of the lines must be similar. The ratio of the larger contrast to the smaller one must be less than a threshold. Alternatively, the similarity of areas adjacent to the lines could also be used. (See Figure 2f).

The same linking process is performed for all line pairs retrieved by the first constraint above. It is applied separately for each pair of endpoints of a given line, and obviously can be done in parallel. The result is a global, directed graph, called the link graph, in which the set of vertices represent all the line segments in the image at a given scale and the arcs represent the links. (See Figure 3a). The linking process effectively reduces the number of combinations of lines that must be examined by the optimization step, which examines paths in this graph and tests them for straightness (see Figure 3b).

Optimization

The second step in the grouping process is the optimization step, which consists of examining all sequences in the link graph in an attempt to find optimum straightline fits and replacing sequences of lines which are linked by a single line. In general, we want to look at as much evidence as is computationally feasible when determining how and whether a straight line should be fit to the data. The amount of geometric context used is determined by the *perceptual radius*, which bounds the length of a sequence of lines that is tested for straightness using a least squares fit. One should note, however, that a very large radius would defeat the purpose of the hierarchical grouping process which is intended to deal with combinatorics of the problem.

Let us consider a given line token in the link graph. Each sequence of line tokens that includes the given line is approximated by a straight line, and if the approximation is sufficiently good, then the sequence will be replaced by a new single straight line token. The algorithm performs the following steps:

1. Generate all paths of lines from the initial point P_1 and all paths from the final point P_2, within the *perceptual radius*, including the paths of length zero.

2. Generate all paths which are combinations of paths from step 2 as follows: a path to the initial point, the line itself, and a path from the final point. A path from one of the endpoints can be null.

3. Fit a straight line, using the least squares method, to the set of endpoints of the individual lines of a path from step 3, and calculate the following measure of straightness:

$$S = \frac{\sum_1^n d_i^2 \cdot l_i}{(n-2) \cdot s^2}$$

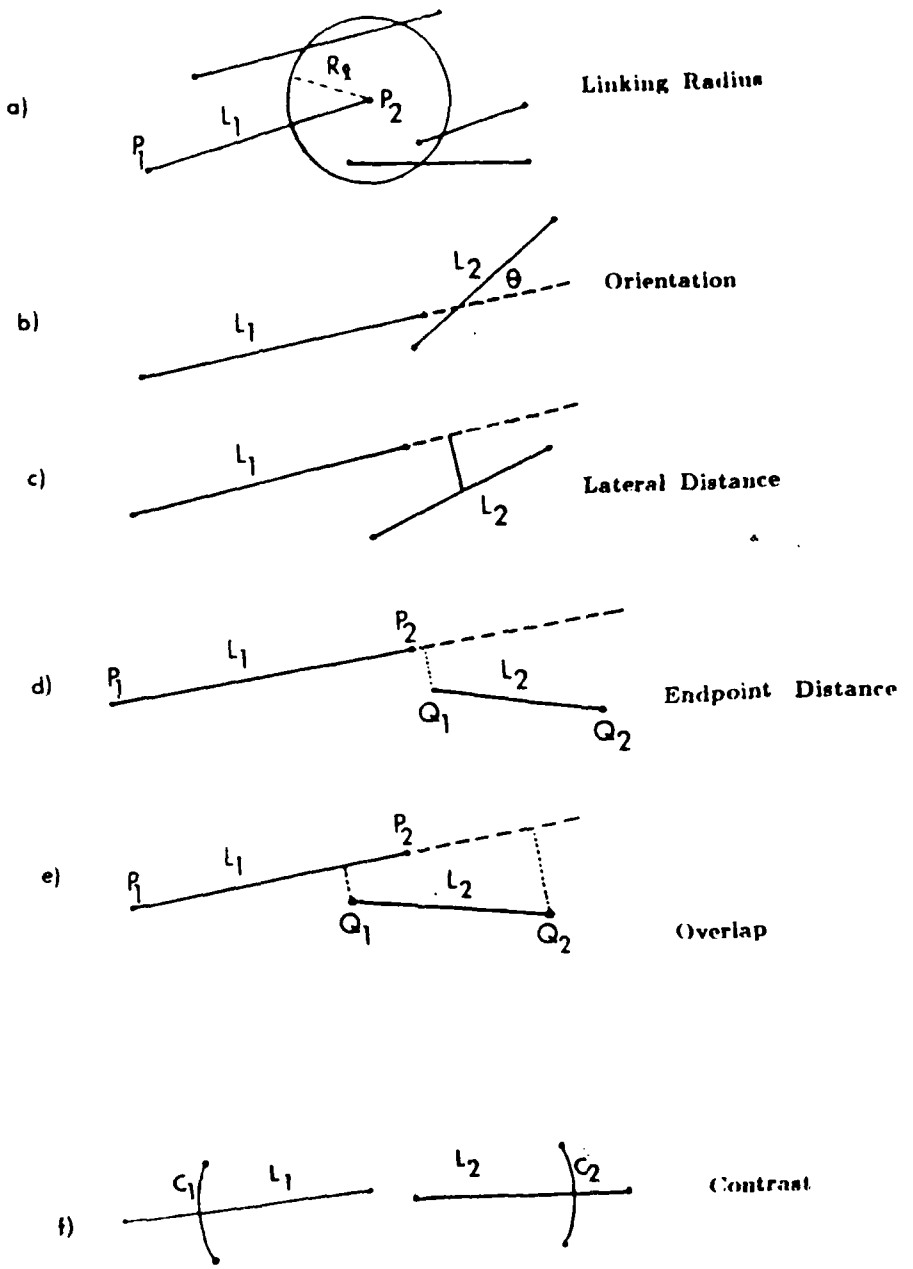
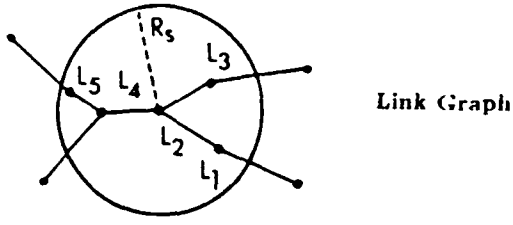
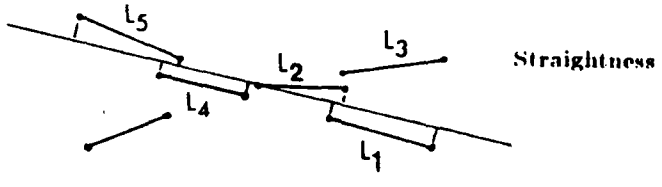


Figure 2: Criteria for linking (a)–(f) and grouping (g)–(h): (a) Proximity: lines are linked with L_1 if they pass within the linking radius R_l from endpoint P_2, where R_l is related to the length of L_1; (b) Orientation: lines must have similar orientation; (c) Lateral distance: lines must be close in the lateral direction as measured by the distance of the midpoint of L_2, perpendicular to the extension of L_1; (d) Endpoint distance: the endpoint P_2 must be close to the projection of the endpoint Q_1; (e) Overlap: the lines must not overlap too much; the distance from the projection of Q_1 to P_2 must be small compared with the distance from P_2 to the projection of Q_2; (f) Contrast: the ratio of the contrast of L_1 to the contrast



(a)



(b)

Figure 3: Replacement.(a) Link Graph: for each line there is a candidate group of lines which are possible extensions from each endpoint; (b) Straightness test: each sequence of lines is tested and the straightest is selected if it passes a threshold test.

where n is the number of endpoints, d_i is the perpendicular distance from the i^{th} endpoint to the approximating line, l_i is the length of the line segment corresponding to the i^{th} endpoint, and s is the scale of the token, which is the distance from the initial point of the first line segment to the final point of the last. The optimum straight line fit on the set of endpoints of all sequences is selected.

4. If selected token sequences has an error less than a straightness threshold, replace the set of line tokens by a new line token whose contrast is the average (weighted by length) of those of its parts.

Replacement and Control of the Grouping Process

On each cycle, each edge or previously grouped line in the link graph is considered for grouping and those sequences which pass the straightness test are replaced by a single token. This is shown schematically in Figure 4. Since the grouping process is iterated many times, some lines will begin to organize into longer lines, while some will remain unreplaced. As new line tokens are formed, new attributes are formed with the contrast computed as an average of its parts weighted by length. Since one would not like to hallucinate lines, there is a requirement that a high fraction of each new line produced be covered by the projection of the matching edges [HER84]. The coverage is computed for each new line.

A key control issue arises if multiple representations involving the same line token are allowed. This can arise in different ways. If the relational constraints that form the link graph are made sufficiently loose, then it is possible that more than one path through the graph is of interest. That is, multiple straight line fits involving the central token (around which the link graph has been formed) could produce very different straight lines, each of which has a low error and may be useful. Thus, multiple low-error line replacements might be allowed if, for example, the lines have very different orientations. This could occur when lines cross at a large relative angle.

If replacement is developed as a local parallel process around each line token, then it would be likely that multiple line fits would involve the same token. Many would be very similar in location and orientation since the link graph around adjacent lines would have paths whose sequence of lines overlap considerably. Thus, token aggregations that are replacement candidates would have to be examined for the presence of tokens in common. The computation could become large and control become complex. Currently a check is made so that if a line is entirely contained in a 1-pixel wide box around a longer line, it is deleted.

We have implemented an algorithm where replacement of a token sequence will take place before any other sequence that involves these tokens is considered. This can be purely sequential, or partially parallel where disjoint subgraphs of the link graph are considered concurrently. In these cases the order of token replacement could affect the result, but we have not found this to be a significant issue.

The hierarchical representation

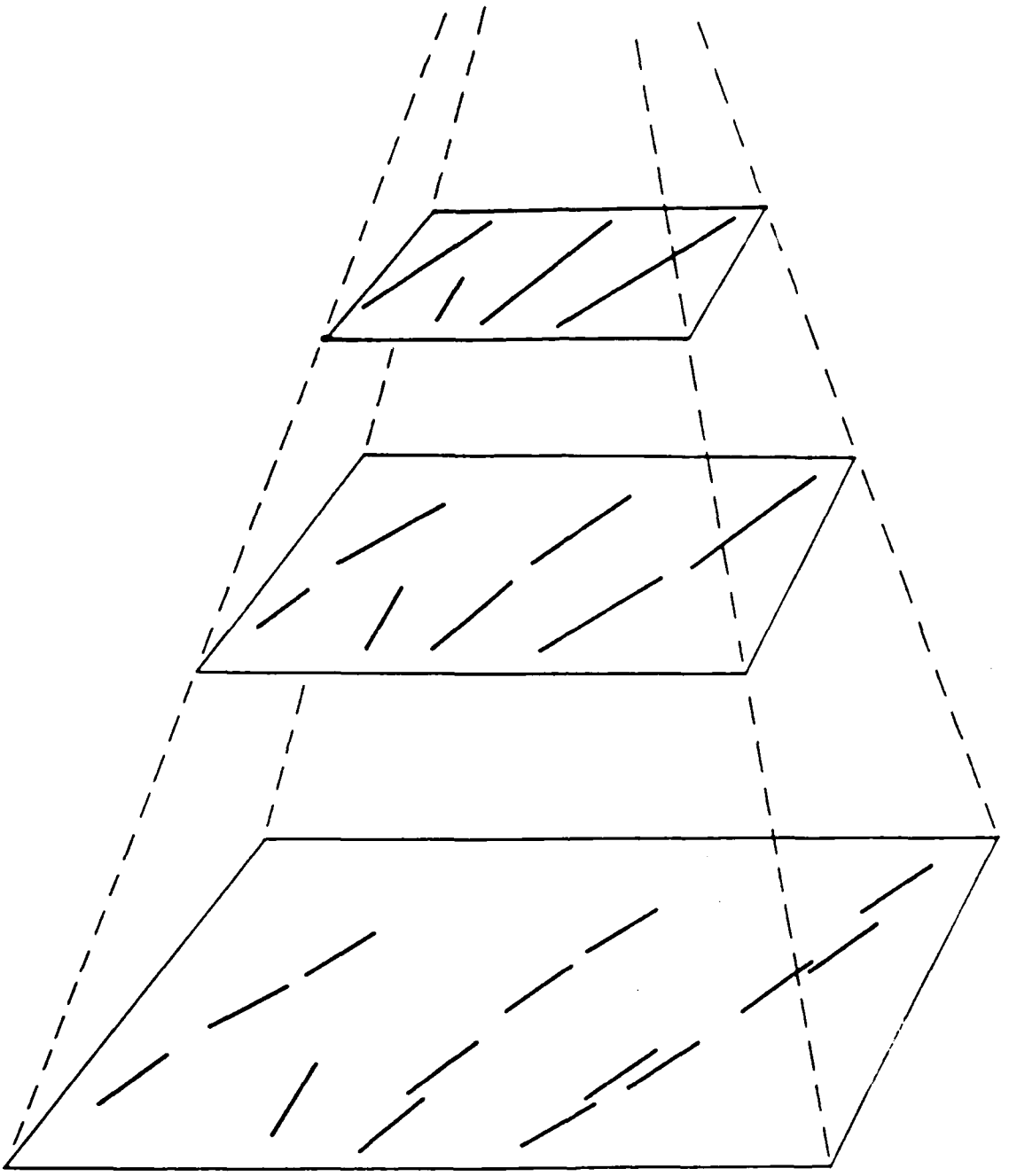


Figure 4: The hierarchical grouping process. Lines at the bottom may be replaced by longer lines at the next larger scale.

The hierarchy used is very simple; it consists of levels, with the scale of the level determined by the range of line lengths contained at that level. There are two important reasons for using a hierarchy. First, the description of straightness is dependent on scale. As was shown in Figure 1, a sequence of lines which is not straight at one scale can be part of a longer sequence that passes the straightness test at a larger scale. Second, there are computational advantages. In our implementation, the hierarchy contains multiple levels, each representing a different scale. A level is a set of directed line segments, represented by endpoints and contrasts. In the current implementation these ranges overlap, and the linking and replacement process is repeated, thereby constructing levels of lines sequentially from lower to higher.

The concept of scale can be described in terms of the grouping process. Edges are computed from a 3×3 window in the image, and this window can be considered as a region of support for the edge [BUR86]. As edges are linked and replaced, the new lines have a support region which is the union of the support regions of the grouped line segments. The diameter of the support region is the scale. The linking and perceptual radii are functions of the scale and are multiplied by constant factors from one level of the hierarchy to the next. The linking radius controls the retrieval of candidate lines from the data base. The perceptual radius controls the length of sequences considered in the link graph. In order to reduce the search computation during the linking step, each level of the hierarchy is divided into a spatially uniform grid corresponding to the density of lines. Ideally, the number of lines will decrease by a constant factor, since a certain fraction of aggregations of line tokens will be replaced by single line tokens. If each line token were merged with another single line token during each cycle, then the number of tokens would be halved; if at the same time the linking radius were multiplied by the square root of two with each cycle, the area would double and, assuming uniform density of lines, the number of tokens to be searched would remain constant. In our experiments a multiplicative factor of 1.2 was used for the linking radii, and therefore we expect the computation to be reduced in later cycles of grouping.

In order to take advantage of the hierarchy, it will be important to have further reduction in the number of the lines as one progresses to larger and larger scales. One might consider removing all lines after each cycle which were not replaced; however, many of them could be grouped in later cycles. Instead under our current grouping scheme, an unreplaced line is copied to the next level up to four times. After that it will be deleted for future cycles of grouping. As a result, lines at a single level will not all have the same length; their lengths will lie within a range, and the ranges for different levels will overlap.

Linking Parameters and Evaluation

As noted above, there are a number of threshold parameters which are used and which, in addition, depend on scale. The parameters that are of major concern are those associated with thresholds on the relational measures to form relations in the linking or filtering phase. If one filters too much, one may eliminate too many good solutions. If one doesn't filter enough, the computational cost will be too great.

In order to select the best parameters for this or any other algorithm, it is necessary to

have criteria for evaluating the results. Two aspects to the evaluation are the quality of the output and the efficiency of the computation. There is often a trade-off between the two measures, and what we have chosen is to find parameters which minimize the computation subject to the constraint that near optimal results be obtained in the output.

In order to evaluate the results of the algorithm that is presented here, a cost function has been defined to measure how well the resulting lines fit the initial edge data that is extracted. We use the average of the mean-square error functions that was used in the optimization phase, and which computes the sum of the squares of the distances between the end points of the data edges and their projections on the line fit to the set of line tokens. However, this cost function would be optimized by allowing no grouping of edges and thereby achieving zero error. Therefore, in addition, a minimum number of significant long lines must be extracted. These lines are selected by hand for each image before the evaluation is carried out. One would expect that there will be critical values of the linking parameters such that changing them beyond some value is inappropriate. For example loosening a constraint might not decrease the mean-square error of the best fit (i.e. a near optimum sequence was already included), but the computation would begin to increase dramatically because there are so many more line tokens included in the link graph. Conversely, tightening the constraint beyond a certain point will undoubtedly prevent the degree of straight line grouping desired (i.e. a desired line will be fragmented). If these critical points in varying the parameters are not identical, then values that balance these tendencies have to be chosen. We have not defined a global cost function across these measures, but that could certainly be done.

The parameters have been chosen so that the resulting cost function on a large variety of images gives reasonable results that are close to optimal and a selected set of significant lines have not been fragmented. Ordinarily one would expect that an algorithm having a large number of parameters might be sensitive to the choice of these values and that their variation would produce a complex behavior. However, the resulting lines from the algorithm here are stable over a wide range of parameter values, which primarily affect the amount of computation. We have determined with some examples that loosening the parameters increased the computation time, but did not significantly improve the quality of the resulting lines in terms of RMS error. For example, doubling the link radius from the optimal value did not change the RMS error, and only 2 out of the 290 resulting lines were affected in a 64x64 image. On the other hand, the computation time was doubled. Increasing the other thresholds on lateral distance, overlap, and relative orientation also did not affect the RMS error, but increased the average branching of the link graph by a factor of three. This means that lines were linked to three times as many other lines as with filtering. In general, amount of computation will increase by a factor which could be very large, depending on the length of sequences tested for straightness as described in Section 2.3.3.

2.4 Experimental Results for Hierarchical Line Grouping

The algorithm has been applied successfully to many natural scene and aerial images. Four of these are shown in Figure 6. We illustrate the algorithm on a subimage of Figure 6a. Figure 7a shows the contours of the Laplacian zero crossings. There are several places where the contour does not follow the boundary of the roof, but wanders off into the texture of the roof or the texture of the trees. Figure 7b shows the initial edge segments which are input to the grouping algorithm. Figures 8a – d show the stages of the geometric grouping algorithm after two, four, and six cycles, as well as the final output by integrating the results at all of the scales by only taking lines from each scale that were not grouped at a higher scale. After the fourth cycle, the outline of the chimney is present, as are large pieces of the boundary of the roof. After the sixth cycle, many of the shorter lines have been dropped. Our geometric grouping algorithm performs very well on the long lines because it integrates evidence from many parts of the image.

Restricting the algorithm to just the subimage containing the chimney, we illustrate linking and replacement in more detail. Figures 9a and 9b show the lines after one and two cycles, respectively, with links shown by circular arcs. In general, only pairs of lines are merged. There are also many cases where linked lines do not pass the straightness test, e.g. the trihedral vertex at the top. There are also cases where merged lines are able to link with lines which were not linked at a smaller scale.

Filtering the resulting grouped lines by contrast or length is likely to retain many of the perceptually significant lines, for example the boundaries of objects in the image. Figure 10 shows the resulting lines whose contrast is greater than 15 gray levels (out of 255); none of the lines in the sky region are left.

As a basis for comparison, Figure 11 shows the filtered output of the Burns straight line algorithm, which we consider to be one of the best straight line extraction algorithms available [BUR86]. That algorithm is very successful in extracting many of the straight lines, but some of the long ones are fragmented. This occurs because the Burns algorithm uses a connected components algorithm to group pixels with similar gradient orientation into an edge support region, and pixel data supporting an edge can be interrupted by aliasing, texture on either side, or even the smallest occlusion. One of the other weaknesses of the Burns algorithm is that the orientation of some lines is distorted because of the way in which their position is determined. A plane is fit to the intensity surface of each edge support region, and the support region may contain extraneous pixels. For example, if there is surface shading perpendicular to an edge, the gradient orientation could remain constant in the shaded and unshaded areas, and if the shading is stronger at one end, the line will be skewed toward the shaded area. This situation does not create a problem for the geometric grouping algorithm because parallel edges do not influence each other and the collinearity constraint would limit the orientation error.

Figure 12 shows the grouping algorithm applied to the entire image of the house scene with lines of length one pixel removed and no filtering on contrast. Removing lines which are short or have low contrast makes it easier to examine the results as shown in Figure 13.

Lines corresponding to large parts of the entire roof boundary, telephone wires, and rain gutters are connected. The only major problem which has been encountered is overmerging of lines, particularly in textured areas where accidently collinear line segments occur (such as in the lines in the roof of Figure 12 and Figure 13 unless the coverage constraint that indicates the extent to which a line is supported by actual intensity variations extracted from the image. If texture is present, then the size of the gaps and their total length compared to the length of the replacement line become important. Thus, lines with gaps (such as a dashed line) which are grouped into a new token will have a lower value of coverage as the size of the gaps relative to the length of the lines increases. Figure 14 shows the effect of filtering by coverage; there is significant improvement in the textured areas, but the problem is not eliminated. To the degree that the coverage relation is more constraining, lines will not be formed in dense texture, but other lines will fragment.

More significantly, overgrouping is a two-dimensional problem, not a one-dimensional one. Lines which are part of a texture pattern will have a high density of similar lines surrounding them, but lines corresponding to a contour boundary often will not. This is analogous to the example in Lowe and Binford [LOW83]: groups of collinear points are difficult to detect when they are embedded in a random distribution of points with the same interpoint distance. Thus, the algorithm could be improved by using the density of lines to inhibit linking when the density is high. In our implementation, the linking radius depends on the scale, but it should also depend on the density of lines. If there is a high density of lines with different orientations, we would only perceive a straight line if the gaps between the fragments were very small. It is interesting to note that smoothing the image with a Gaussian mask has the effect of removing most of the texture. This can be seen in the roof of the house in Figure 15. Since the lines have low contrast and a high density, when the image intensity is smoothed, the lines interact and the contrast is further reduced. However, it is also clear in this image that significant structure is also removed by smoothing.

We have performed a few tests to examine the effects of filtering to remove very low contrast lines, which might be related texture (note, however, that these lines can serve as a texture measure [BUR86,HAN87]). Figure 16 shows the resulting lines after filtering low contrast lines of the road scene. Figure 17 shows results for the other images in Figure 6.

2.5 Generalization to a Symbolic Grouping Process

In order to extend our framework of perceptual grouping of straight lines to include such events as texture and complex contours, it will be necessary to consider more complex tokens as well as grouping and abstraction processes. A plausible token vocabulary might be:

1. points (dimension 0): spots, corners, line endpoints, etc.;
2. lines (dimension 1): straight line segments, circular segments, quadratic curve segments, linked line segments, etc.;
3. areas (dimension 2): closed curves, parallel lines, rectangles, general polygons, texture patterns, etc.

If one were to approach this problem via a generalized Hough transform [1,2] each class of geometric structures would need to be recognized separately. Considering the issues mentioned earlier of the loss of spatial and geometric context, we believe that more general symbolic grouping processes are needed.

There are two types of operations which we would need for our framework of symbolic grouping. The first type involves forming token aggregates which are then represented by a single token; this is called "grouping". The second type occurs when we want to ignore some of the information present in a token; this is called "abstraction". Abstraction not only reduces the amount of information which needs to be stored for a token, but also allows a simpler grouping process to be applied. For example, an area which is not elongated might be reduced to a point; several areas could then be grouped by a point-grouping process. An area which has a strong major axis can be reduced to a line which approximates that axis; e.g. smoothed local symmetry [BRA84] or medial axis transform [BLU73]. A line which has a short length can be reduced to its midpoint. In all of these cases, the criterion for the abstraction process will depend on the relationship between the size of the token, the scale at which processing occurs, and the goals of the grouping process. As a more complex example of hierarchical grouping and abstraction, consider Figure 5 and Table 1.

A general system which allows all possible abstractions will require a sophisticated control structure to decide which abstraction to apply and when. If one were simply to allow all abstractions and groupings to be instantiated, the number of tokens would increase rapidly, resulting in a large increase in the processing time. It seems clear that of some grouping processes are sufficiently general that they would be useful in almost all domains, e.g. straight line grouping; it also seems likely that in other cases top-down strategies would be useful as a way of limiting the types of grouping.

Let us assume that the most primitive detector finds gradients in the image, and the token vocabulary consists of circular areas (spots), points, line segments (circular or straight), and closed curves (circles). Some tokens represent a complete unit or 'whole' in that they can be considered as primitives, i.e. they are not part of a larger aggregate

Token Abstractions		
gradients	group to	an area (spot) ('whole')
an area	reduces to	a point
points	group to	a curve segment
curve segments	group to	a circle (area) ('whole')
a circle	reduces to	a point
points	group to	a line ('whole').

Table 1: Example of grouping and abstraction for Figure 5.

Grouping (takes multiple tokens to one token)		
dimension of input tokens	dimension of output tokens	description
0,1	1	points or lines group to a line
0	2	points group to an area
1	2	lines group to an area, e.g. texture, rectangles etc.
2	2	areas group to bigger areas

Abstraction(takes a single token to another one)		
dimension of input tokens	dimension of output tokens	description
1	0	small line is replaced by a point
1	1	rough line is smoothed
2	0	small region is replaced by a point
2	1	elongated region is replaced by a line

Table 2: Summary of token grouping and abstraction.

without first undergoing abstraction. A general set of transitions for the token vocabulary above is shown in Table 2.

In this paper, we have discussed grouping of line tokens which are of the same type; however, work on grouping of tokens of different types, and the grouping of tokens of the same type to produce a token of a different type has also been started. Riseman *et. al.* [RIS87] have examined regions and lines together in order to combine the information available from different types of segmentation algorithms. This can be viewed in terms of the above formalism as creating an area token that includes in its description the attributes of the regions and lines grouped together, as well as several relational measures based on their intersections. Reynolds and Beveridge [REY87] have grouped aggregates of spatially proximate parallel, collinear, and orthogonal lines into area tokens and have produced very useful results. They also discuss the problem of deciding which aggregates

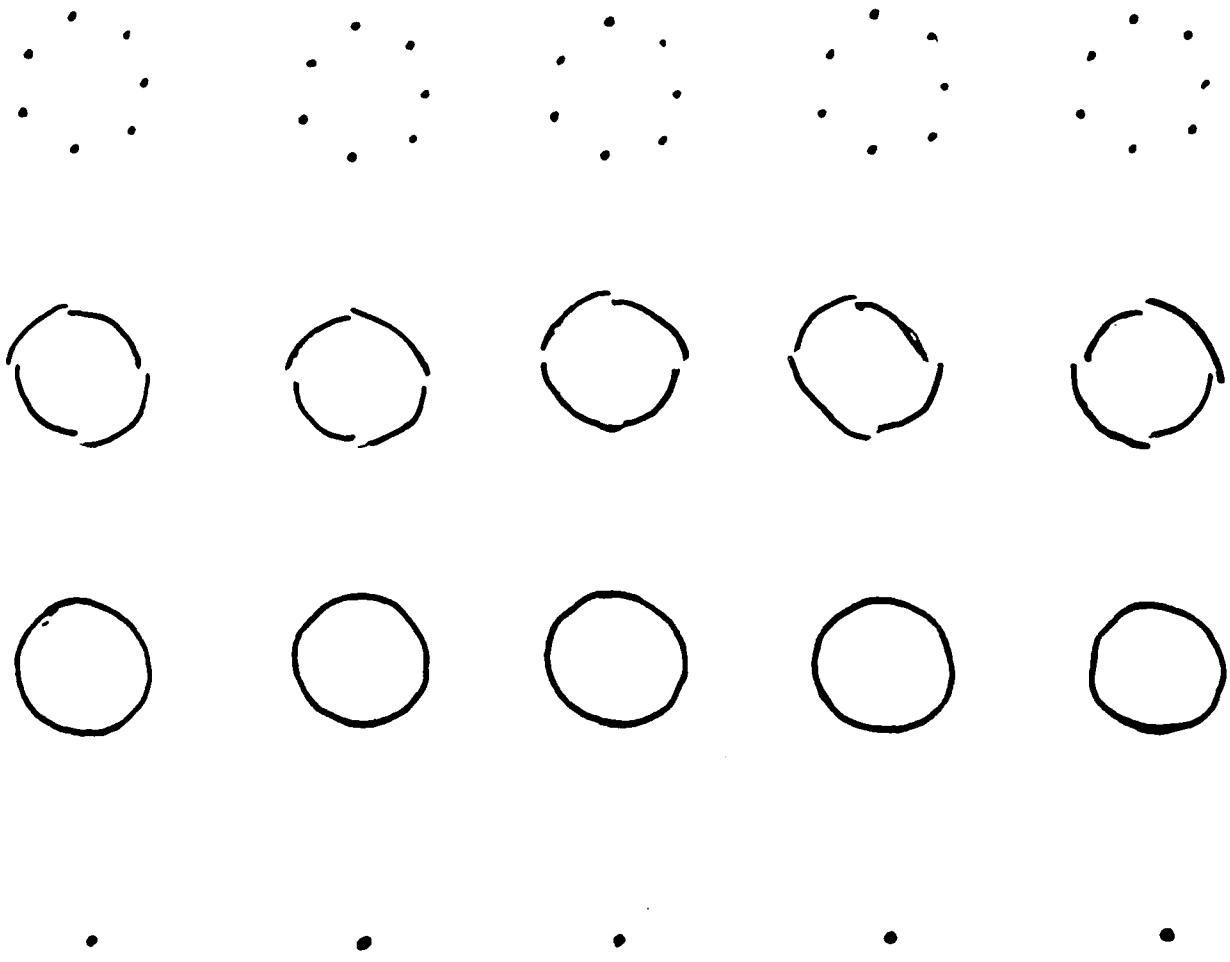


Figure 5: An example of a more general abstraction process: points are grouped into curved lines, curved lines are grouped into closed curves that define areas, areas are reduced to points, and points are grouped into a straight line.

are most significant.

2.6 Discussion

We have outlined a hierarchical approach to symbolic grouping. Locally optimal straight line fits to subsets of tokens are determined by using relational constraints of proximity, collinearity, and similarity of contrast. Our methodology shows that heuristic rules for filtering based on these constraints are effective in reducing the computation of extracting straight lines. The hierarchy is used to represent objects at different scales, and each level of the hierarchy has its own appropriate geometric context. The hierarchical application of these constraints allows the search for the proper token aggregations to be computationally feasible while integrating information over the entire image. The algorithm has been tested on a wide variety of images, and the results indicate very good preformance in extracting straight lines from complex images.

Since token-based grouping has been successful in this particular application of straight lines, it is a good candidate for generalization to more complex and abstract geometric structures, and we have outlined such an approach. It is currently being applied to the simultaneous extraction of straight lines, curves, corners, and cusps [DOL89]. There are also ongoing efforts to extract rectilinear and polygonal structure via the linking, optimization, and replacement cycle of grouping[BEV89.REY87]. Williams and Hanson[WIL88] have applied this paradigm to grouping lines in the temporal dimension. This has made it possible to track lines from moving objects over a sequence of images, even when the lines undergo fragmentation and merging. Finally, it seems likely that top-down strategies will be necessary to limit the types of grouping and abstraction which are applied, and this also is under investigation [DRA89].

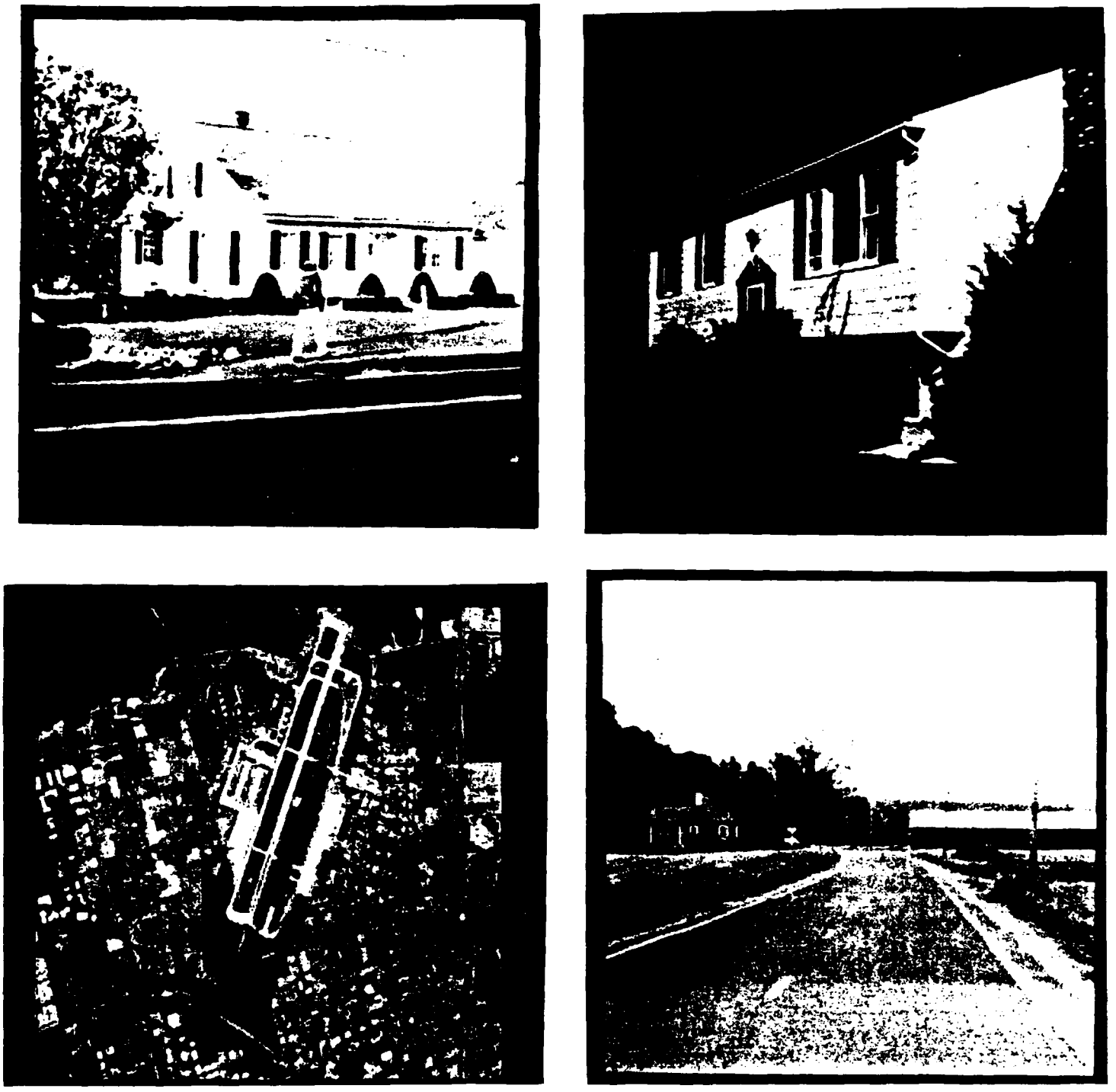


Figure 6: Digitized images of a) house scene, b)house scene, c) aerial image of runway, d) road scene.

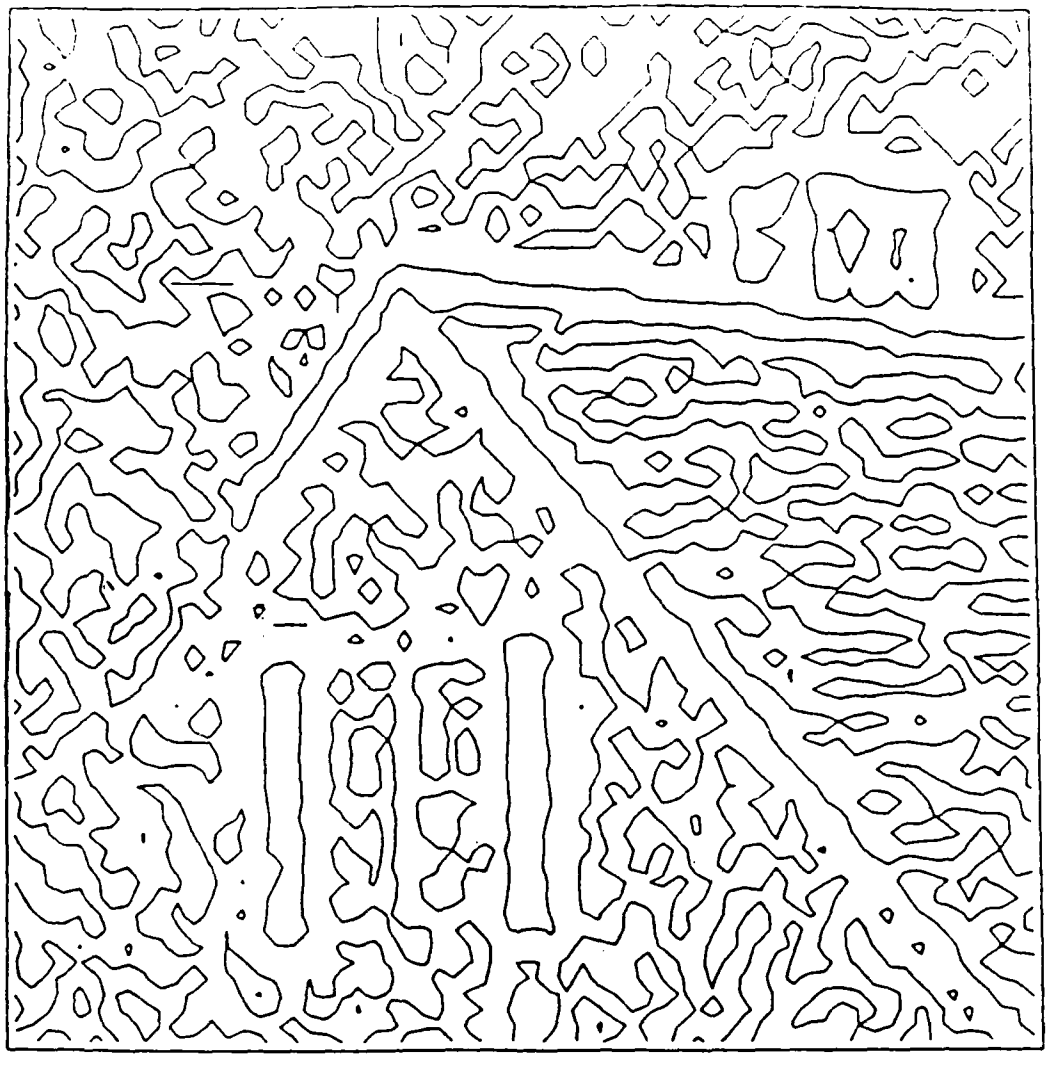


Figure 7: Zero crossing contours of the Laplacian for part of a house scene. Many of the boundaries are fragmented.

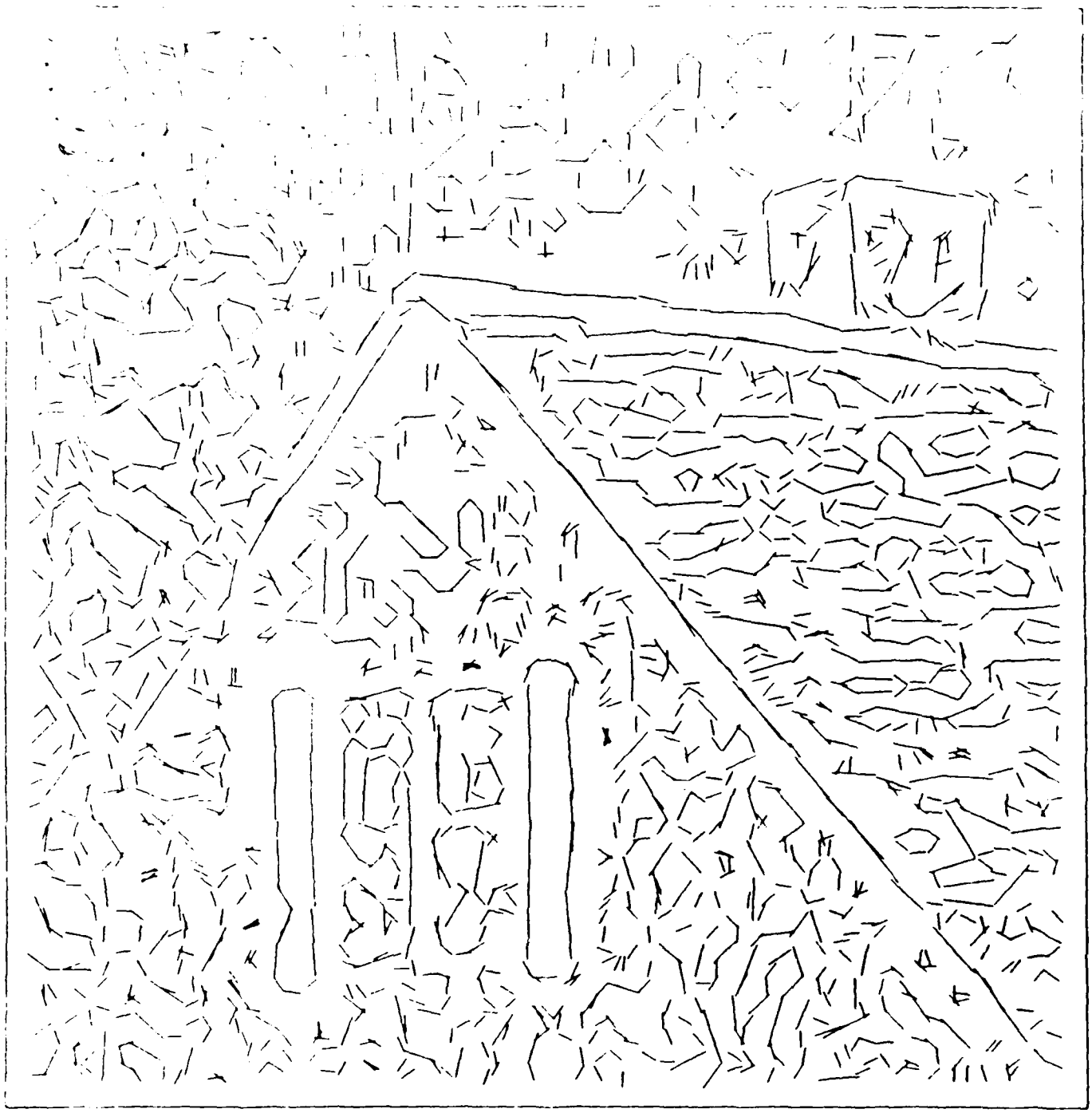


Figure 8: Grouping algorithm after two cycles.

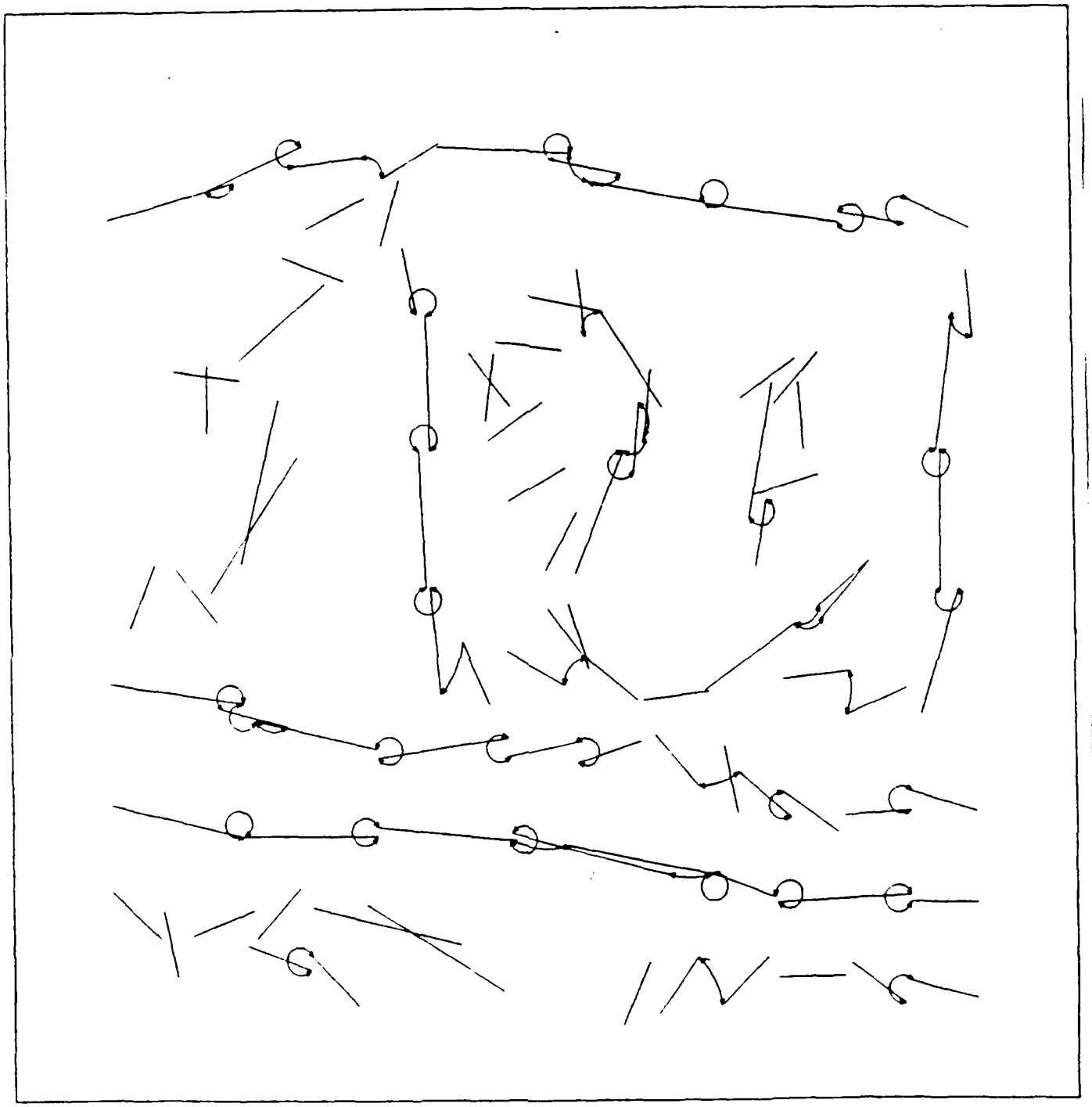


Figure 9: Chimney subimage: the linking process is applied to the output of the first cycle. Links are shown as circular arcs.

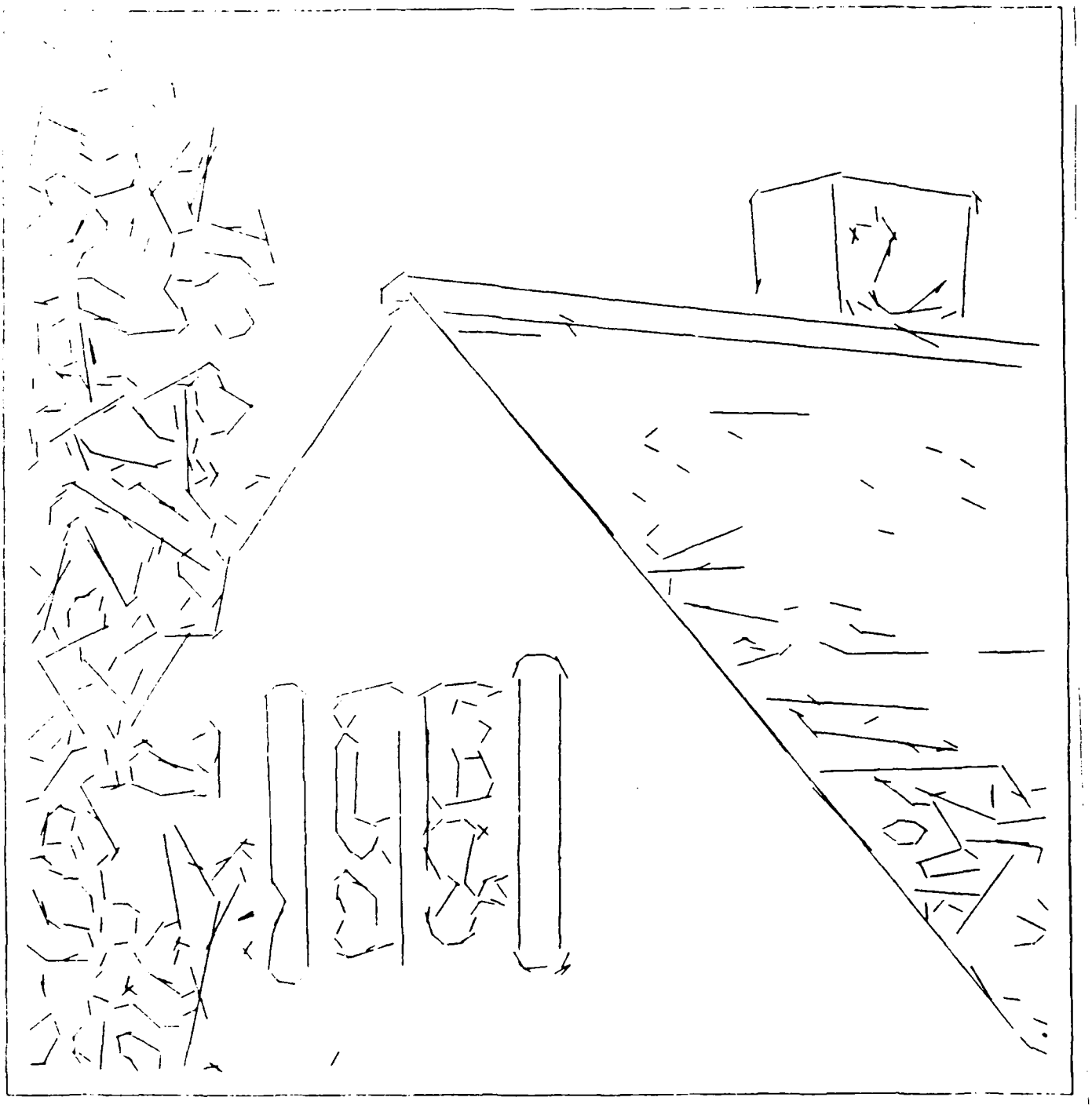


Figure 10: Filtering on gradient magnitude removes low contrast lines. The threshold was set at 15 (for a range of 255).

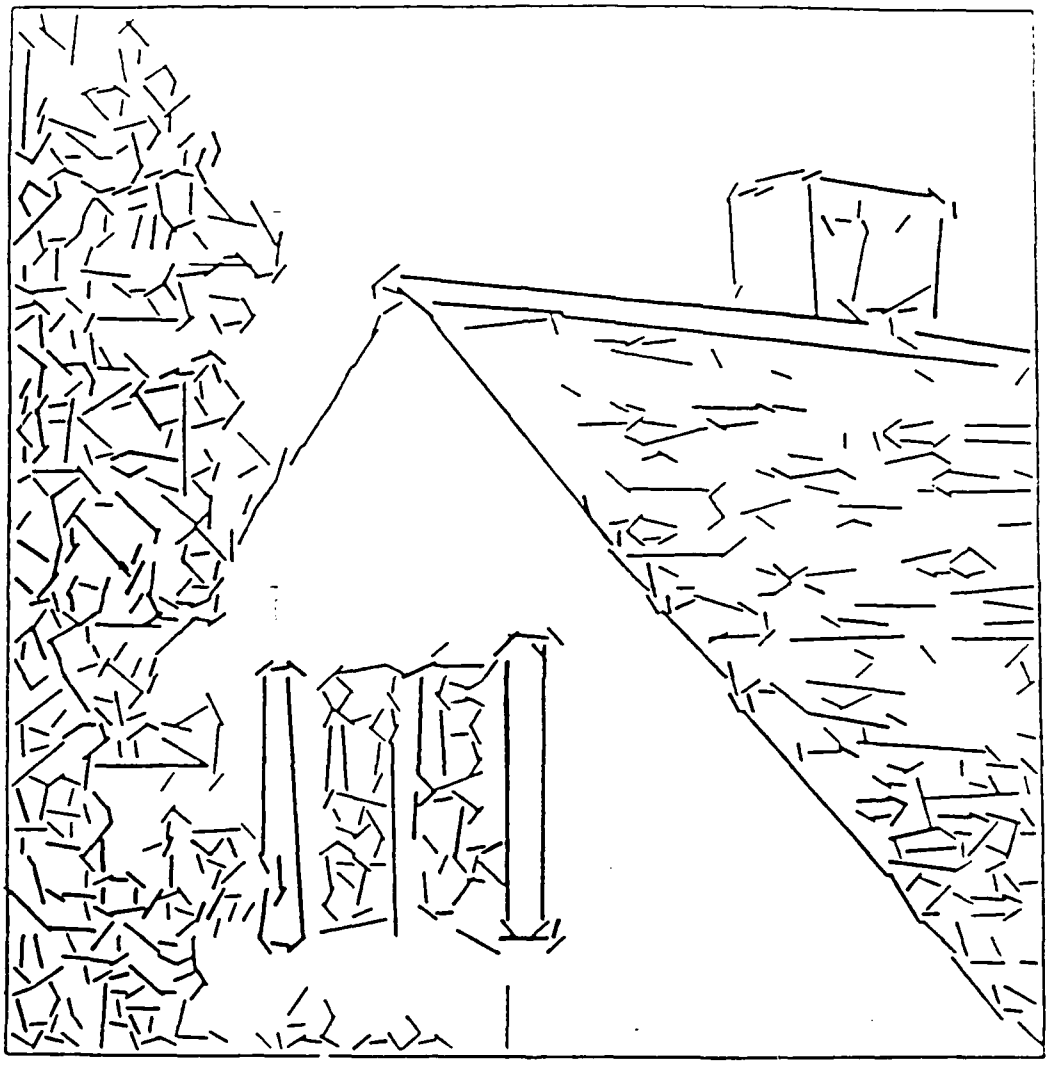


Figure 11: Output from Burns straight line algorithm, filtered on contrast. The locations of lines is similar to the grouping algorithm, but there is more fragmentation.

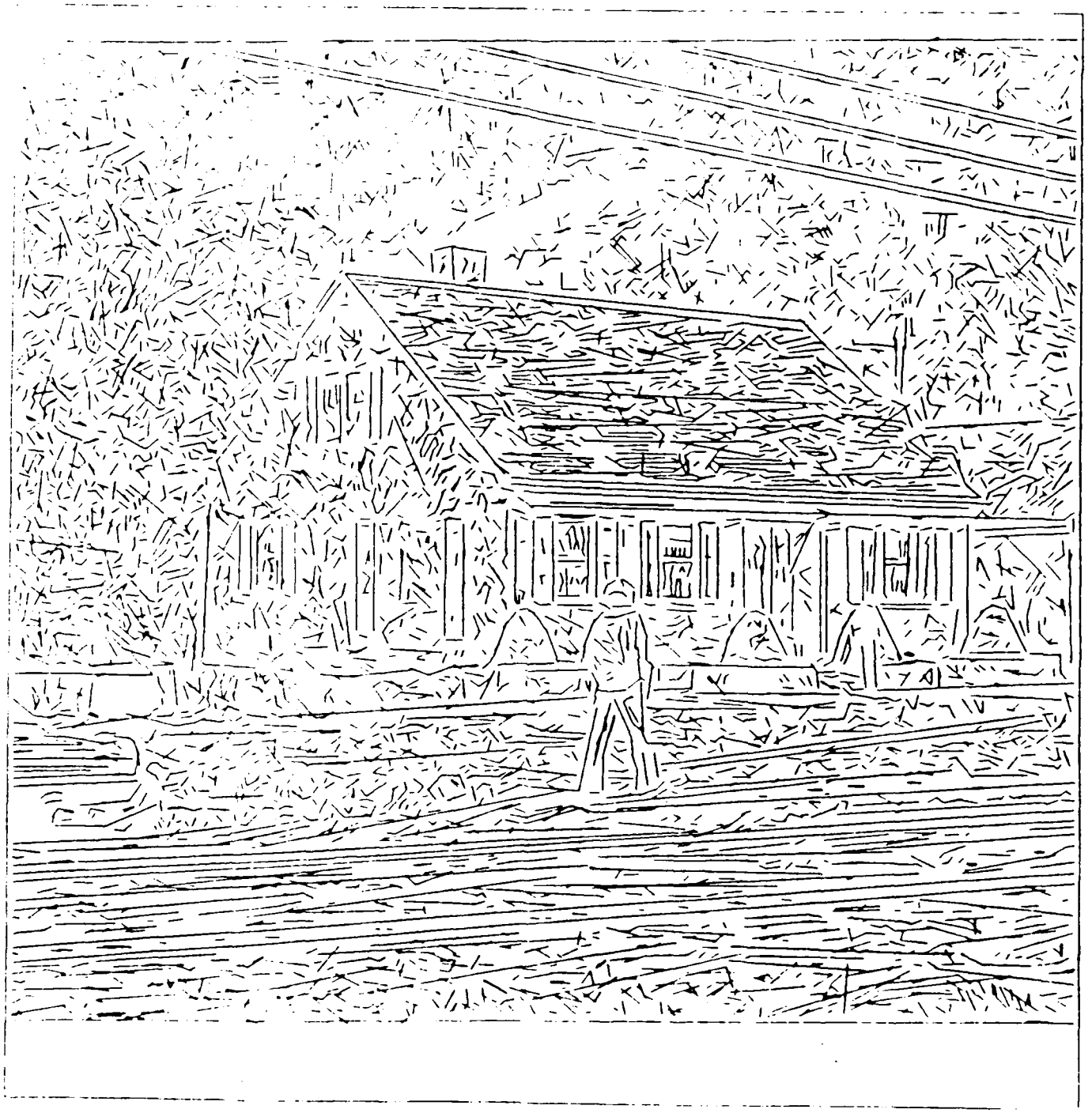


Figure 12: Final output of grouping algorithm on entire image. Lines of length one have been removed. Lines at all scales are shown.

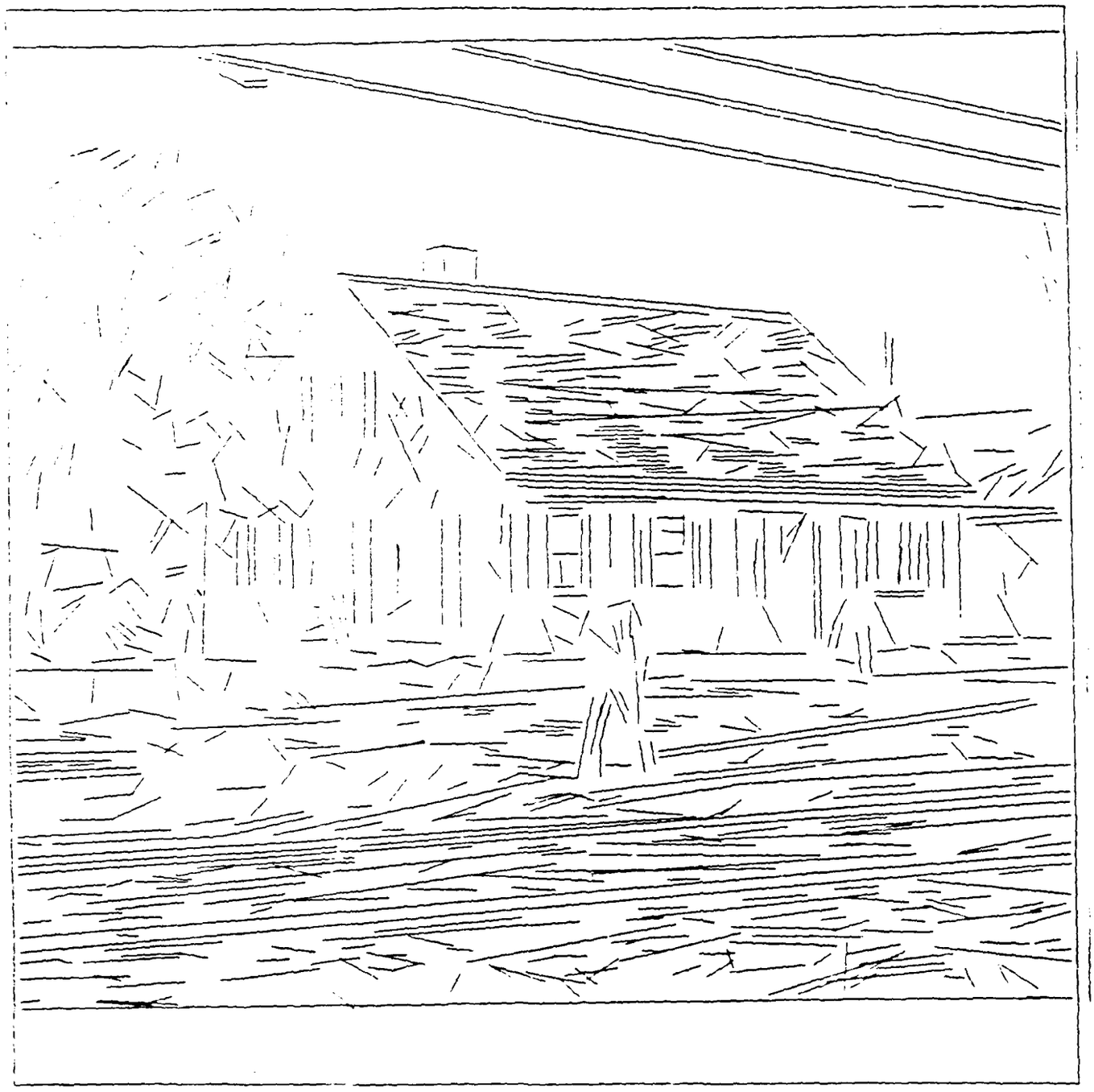
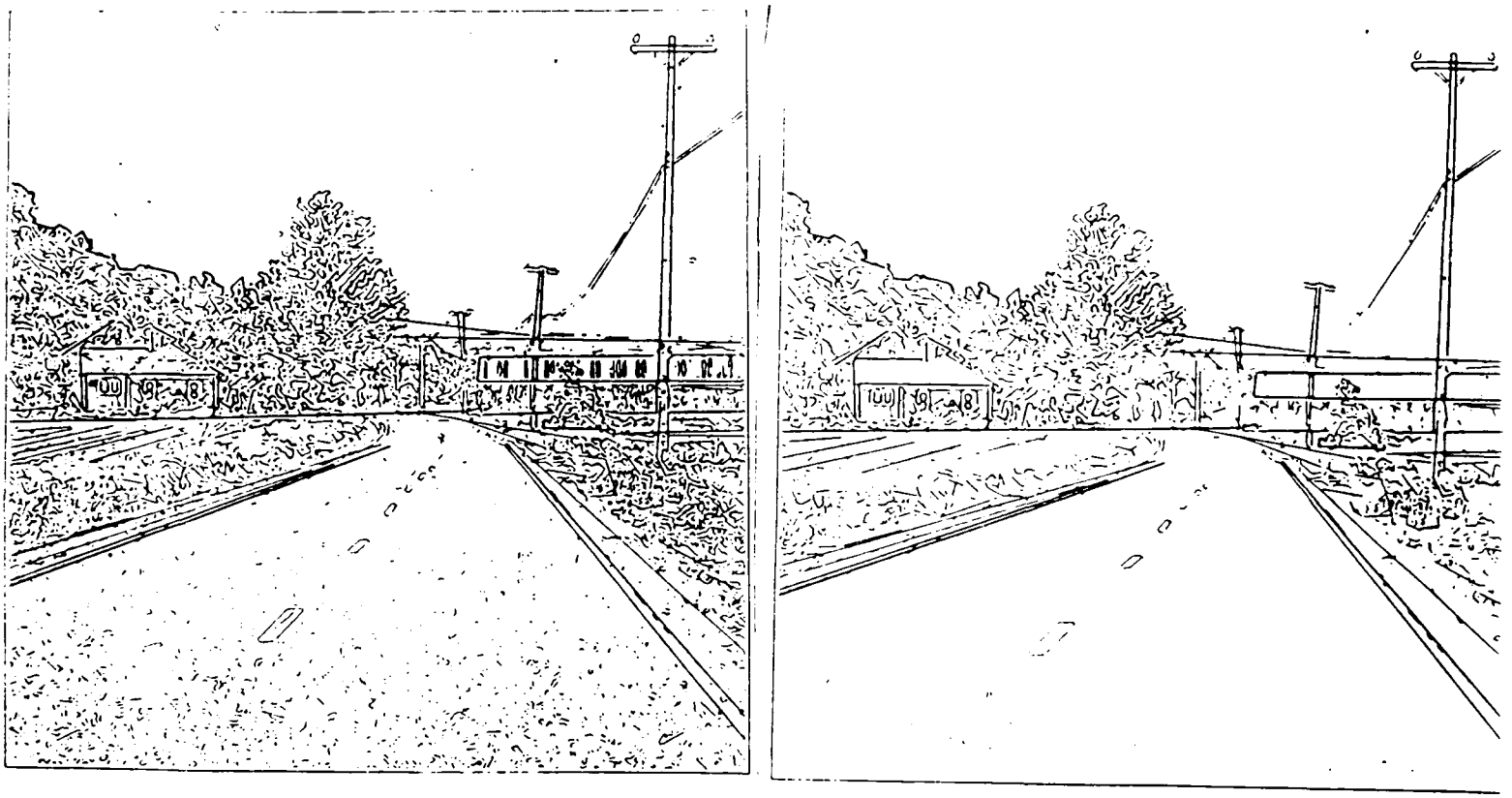


Figure 13: Final output of grouping algorithm on entire image. Lines of length less than 5 and contrast less than 5 have been removed. Lines at all scales are shown.

Figure 14: Lines from the house scene with minimum coverage of 90 percent. Lines of length less than 5 and contrast less than 5 have been removed.

Figure 15: Final output of grouping algorithm on image which has been smoothed by a gaussian with $\sigma = 1.5$. Lines of length less than 5 and contrast less than 5 have been removed. Lines at all scales are shown.



(a) (b)

Figure 16: Lines in the road scene. a) Filtered to remove all lines with contrast less than 5 (out of 255). b) Filtered to remove all lines with contrast less than 10. Note that contrast is measured by average gradient magnitude, and lines with contrast greater than 30 are shown thicker.

Figure 17: Output of the geometric grouping algorithm for house image in Figure 3, filtered on length.

3. The View Sphere for Curved Surfaces

Finding a representation of three-dimensional shape which is both concise and complete has important applications in computer vision. Every surface has a number of distinct typical views. We construct a model based on these views and the transitions between them. The result is a labelled graph on the viewing sphere. The label for each face is itself a graph which qualitatively represents the outline of the object when seen from any point on the interior of that face. For a generic smooth surface, the singularities which occur in typical views, as well as all transitions between those views, can be described by a known finite list of possibilities. We have worked out the models for a representative selection of examples of generic and non-generic surfaces, including a sphere with a bump, a rotationally symmetric and a twisted torus, and a surface with a furrow.

3.1 Introduction

One of the open questions in image understanding today is how to use the shape of an object as one of the features by which it can be identified from an image. While there are numerous techniques for representing three-dimensional objects, there does not seem to be a general method for representing surface shape [BES86]. Our approach to this question assumes that shape description should be based on the information which is contained in the variety of viewpoints from which an object is seen. To begin, we have concentrated on the apparent contour, or outline, which can be extracted from each of these viewpoints. We have thus not yet incorporated information about surface orientation (which might be deduced from shading or range data).

An object could be represented by all of its two-dimensional images as seen from a set of predetermined closely-spaced viewpoints uniformly distributed around the object. This representation can be made as comprehensive as we wish by packing the viewpoints sufficiently close together. However there are several drawbacks. First, the representation would be inefficient both in the space required to store a significant number of views and in the time required to search for a match with a given image. Of course these problems are multiplied as the number of objects to be represented is increased. Second, the distribution of viewpoints takes no account of the shape of the object: in some directions, the image of the object may change dramatically with a very small change in viewpoint, while in other directions there may be scarcely any change in the image over a wide range of viewpoints. Third, this approach is not independent of scale.

The disadvantages of this straightforward approach led us to search for some other way to represent an object by two-dimensional images of its surface. To be more useful, an alternate approach would have to be more economical—minimal, if possible—and tied to the geometry of the object, while remaining comprehensive. The model for describing surface shape which we propose is intended to achieve these goals. It is comprehensive in the sense that it contains an example of each topologically distinct image of a surface, and minimal in that each distinct image appears only once. Finally, the viewpoints for

these different images are determined by the geometry of the surface, and the complexity of the representation reflects the geometric complexity of the surface. For example, a highly symmetric surface will have a simple description.
There are at least two ways in which this method of geometric representation might be applied. First, it serves as a tool to identify objects by matching them with models. Second, it offers a description of shape from which geometric properties such as the sign of the curvature could be deduced. Such a description could be based on current work on two problems: first, to construct a model of an object from a mathematical description; and second, to extract a mathematical description of the outline and its characteristic features from a real image.
The model we present here is derived from certain recent developments in the mathematical theory of singularities of maps [ARN83,KER81,MCC80]. We wish to acknowledge a special debt to the pioneering work on shape description done by Jan Koenderink and Andrea van Doorn [KOE76].

3.2 Viewing a surface

For the sake of definiteness, we assume the object we are considering has a smooth surface and is transparent. In another paper we shall describe modifications needed to handle *smooth opaque surfaces*. *Non-smooth* surfaces (such as ordinary polyhedra) are a different matter. Some of the ideas carry over readily enough to this case, but the mathematical theory which underlies our approach assumes smoothness, so no general results are available.
Let M denote a smooth surface in R^3. For any unit vector v in R^3, let L_v denote the plane in R^3 which is orthogonal to v, and $P_v : R^3 \rightarrow L_v$ the projection parallel to v. Then P_v projects M and each of its tangent planes to L_v. See Figure 18. We say x in M is a *regular point of* P_v if P_v projects the tangent plane to M at x onto L_v, and a *singular point of* P_v otherwise. Generically, the singular points of P_v form a curve Σ_v on M. As it happens, many of the results in this field do not hold categorically but only in typical, or generic, situations. In the present case, "generic" means "for almost all directions v". We shall use the term frequently, in various contexts, in what follows. The image $C_v = P_v(\Sigma_v)$ is called *the contour of* M *in the viewing direction* v or, more simply, *the view of* M *from* v.
There is a view of M from each point v on the unit sphere S^2. We shall call S^2 *the viewing sphere* because the view of M from any point v is what the eye would see if it were placed at v and a small, scaled-down copy of M translated to the center of S^2. It has been shown that a generic view of M is a union of curves whose only singularities are cusps and crossing points where two branches of the contour are transverse.
Furthermore, a generic view is one which, *by definition*, persists over an open set on the viewing sphere. This means that if C_v is a generic view and w is a point on the viewing sphere sufficiently close to v, then C_w and C_v "look alike". More precisely, there is a smooth map $L_v \rightarrow L_w$ with smooth inverse which carries C_v to C_w. While on any single

open set all views are qualitatively the same, between two adjacent ones the views must differ qualitatively. That is, the number of cusps or the number of crossing points in the view C_v must change as the viewpoint v traverses the boundary between one open set and the other. The complete boundary, which is the complement of the union of the open sets, consists of a number of arcs meeting at vertices; in other words, the complete boundary has the structure of a graph.

By definition, no view from a boundary arc or vertex is generic; however, all views along a single arc look alike, just as they do on a single open set. Each boundary view represents a particular transition between generic views. For arbitrary surfaces, there is no limit to the number of distinct transitions. However, if the surface comes from the special class of what are called *generic* surfaces, then only certain transitions are possible. They are only a few dozen in number, and have been completely catalogued within the last decade [KER81]. The technical definition of a generic surface need not concern us; roughly speaking, it is one whose shape has no special features. We must keep in mind the distinction between a generic *view* and a generic *surface*, though. As we shall see in the examples below, a non-generic surface can have generic views; these are just the views (of any sort of surface) which persist over an open set on the viewing sphere. On the other hand, a generic surface can have non-generic views; these are just the transitions between different generic views.

We can illustrate many of the ideas in the preceeding paragraphs by a simple example. An ordinary torus has only three distinct generic views; they are labelled *a*, *b*, and *c* in Figure 19. (View *d* is special; we shall discuss it later on.) The open set on the viewing sphere from which each of these views is seen has been labelled the same way. Note that, because we are assuming the torus is transparent, antipodal points on the viewing sphere share the same view. In particular, each zone carrying a generic view in the northern hemisphere is copied exactly in the southern hemisphere.

As we can see, the only singularities in the generic views are cusps and crossings. Of course, the contour may be completely smooth, as in the "polar" view *a*. The transitions from *a* to *b*, and from *b* to *c*, happen to come from the standard catalogue. They are called the *swallowtail* and the *tangent crossing*, respectively. The rotational symmetry of the torus carries over naturally to the viewing sphere. One consequence is that there can be no transitional *vertices*. The boundary graph (which separates the open regions of generic viewpoints) consists entirely of disjoint closed loops which appear as "parallels of latitude" on the viewing sphere. Another consequence of rotational symmetry is that the swallowtail transition always occurs simultaneously at two different points on the contour; hence the label *double swallowtail*. We should therefore consider that, in this case, the corresponding transition curve on the graph consists of a pair of closed loops, one lying directly on top of the other.

Finally, we turn to the equatorial view *d*. When seen from any point near the equator, our torus has a "front half" and a "back half". As Figure 20 shows, what makes *d* so different from the other transition views is that the singular points from the front and back halves appear to fuse together when the eye moves to the equator. Of course we

must expect different parts of a contour to cross at one or several points, and they may even be tangent at some of those points. Entire segments will coincide, however, only if the surface has a very special shape, that is, only if the surface is non-generic. So the ordinary torus is non-generic. Actually, its shape is special in two ways: first, it is rotationally symmetric, and second, a plane placed on top of it will be tangent to it along an entire curve. We can make the torus generic by distorting it, as in Figure 20. This breaks the symmetry and guarantees that no plane can be tangent to it except at isolated points. The special view d disappears, but many new ones crop up. We shall get a glimpse of them when we look at the viewing sphere of a generic torus in the final section.

3.3 The model

All the elements of our model for describing surface shape have now been introduced. Beginning with any surface M—generic or not—we can associate a graph on the viewing sphere. The faces of this graph, which we call the *viewing graph of* M, are the open sets which carry generic views of M, and its edges and vertices are viewpoints which carry the non-generic transitional views. We define the *viewing data of* M to be its viewing graph taken together with a view from each face, edge and vertex. Figure 19 gives the viewing data of a rotationally symmetric torus. *The viewing data of a surface is the model we propose to use to describe the shape of that surface.*

Viewing graphs are readily adapted to computer representation, and the individual views can also be treated as one-dimensional graphs. The representation can be simplified in certain ways if the surface is generic, as follows: since the transitions which can appear in a view of a generic surface are all described and named in a catalogue of standard forms, we can discard each view associated with an edge or a vertex of the viewing graph of M (i.e., each transitional view) and simply label that edge or vertex with the catalogue name of the transition which occurs there. (In Figure 19, for instance, the double swallowtail and the tangent crossing views are adequately described by their labels alone.) This is not to say, however, that there is a reduction in the viewing data when a non-generic surface is made generic: although the non-generic torus can be adequately described by only six views, including all the transitional ones, its generic cousin needs many more, even if we count only generic views.

Our model, whether applied to a generic or a non-generic surface, has the following features: first, it gives a complete two-dimensional description of a surface in space, in the sense that it contains a copy of every distinct view of the surface; second, it gives a minimal description in that it does not duplicate views which look alike, i.e., which have identical qualitative features; third, it is structured to reflect the shape of the object being viewed. The first two features should be clear from the discussion up to this point. We will try to illustrate the third feature in the following section.

3.4 Some examples

Example 1: A sphere or a convex body.
A sphere looks the same from every direction. Its viewing graph therefore has only a single face, carrying this one view, and no edges or vertices. A convex body is qualitatively the same as a sphere; every view of it is a smooth closed convex curve.

Example 2: A bump on a sphere.
Raise a slight bump on a sphere. By this we mean distort the surface so that the curvature becomes negative on a small annulus. The bump is this annulus together with the small inner disk where the curvature stays positive. If the bump's cross-section is a circle, then the new surface still has rotational symmetry around a vertical axis, and it is non-generic. Its viewing data is shown in Figure 21. We draw only the northern hemisphere because the southern is a mirror copy. Also, for simplicity we draw only the generic views. The non-genericity is reflected in the double swallowtail transition. If the cross-section of the bump is now elongated, the surface becomes generic, two new generic views are produced, and the augmented viewing data is shown in Figure 22.

Example 3: A bump on an arbitrary surface.
The pattern of rings we see running from the north to the south pole of the viewing sphere in Figure 22 (viz.: lip, tangent crossing, beak-to-beak, braided swallowtails; repeated in reverse order in the southern hemisphere) is characteristic of a bump, no matter where it appears. This means that if a generic bump is raised on a positively curved part of an arbitrary surface M with viewing graph G, the new viewing graph will just be the set-theoretic union of G and the viewing graph of the generic bump shown in Figure 22. We are referring here to the boundary elements of the viewing graph, i.e., its edges and the vertices. Concerning faces and views, we can say the following: Each face f of the new graph will be the intersection of one of the faces of G with one of the faces of the bump's graph, and the view from f will be a montage of the views (of M and the bump, respectively) carried by the faces whose intersection produced f. We can summarize these observations by saying that *viewing data is "additive"*.
Additivity creates a problem: the viewing graph of an object can be very complicated because the surface has many small bumps. One solution to this problem is to create a hierarchical representation based on spatial resolution. In order to use this representation to identify objects in real images, it will be necessary to solve a few problems:

1. For each image the contours must be extracted, and the singularities located.
2. For contours which correspond to multiple views of the same surface, singular points must be matched.
3. A high level viewing graph must be constructed from the multilple views
4. The high level viewing graph must be matched with a subgraph of the viewing graph of the model.

It should be noted that the analysis need not proceed bottom-up from 1. to 4. For example, one could look for cusps or crossings in a particular configuration which lie on the same contour component. Also since there are only a finite number of types of vertices in the high level graph, i.e. transitions, it is not necessary to be completely accurate in steps 1 and 2 in order to get the correct result in step 3. In addition, since there are several types of singularities, this will restrict the search in step 4.

Example 4: A furrow on a convex body.

This example was introduced by Koenderink and van Doorn [KOE76]. Recently they extended their analysis of the furrow to include what we are calling its viewing data. They have described some of their results to one of us; this is a report of that communication.

A furrow is, like a bump, a particular sort of intrusion of negative curvature into a positively curved region. The patch of negative curvature on a kidney bean or a sausage is a furrow, but perhaps the best way to visualize one is through its viewing data. This is given in Figure 23. Notice the distinctive hour-glass shape of the viewing graph; it is characteristic of the furrow, and will appear, by the additivity principle mentioned in the last example, in the viewing graph of any surface which has a furrow. The new vertices which appear on the viewing graph are the *godron* ("*ruffle*") and the *gouttiere* ("*gutter*"). They are found in the standard catalogue and are also known there by the names *gulls* and *goose*.

Example 5: A generic torus.

We illustrate in Figure 24 the viewing graph of a generic torus. It is quite complicated when compared with the non-generic torus (Figure 19). The double swallowtail transition is resolved into a braided pair of single swallowtails, just as it was in the bump. The equatorial view *d* (Figure 19) is also resolved, using the *godron* vertex which appeared in the viewing graph of a furrow.

A further link between the geometry of a surface and the structure of its viewing data can be found in the last three examples. *Every* region of negative curvature on a generic M must reveal itself in the viewing graph by beak-to-beak and lip transitions along certain edges which form the complete boundary of an open region on the viewing sphere.

3.5 Summary

What we have tried to do here is to present a representation of surface shape in such a way as to show its relevance to computer vision. While several problems need to be solved before this representation can be used in practical applications, it has the advantage of being concise and complete, in the sense that it contains all the qualitatively different views of an object.

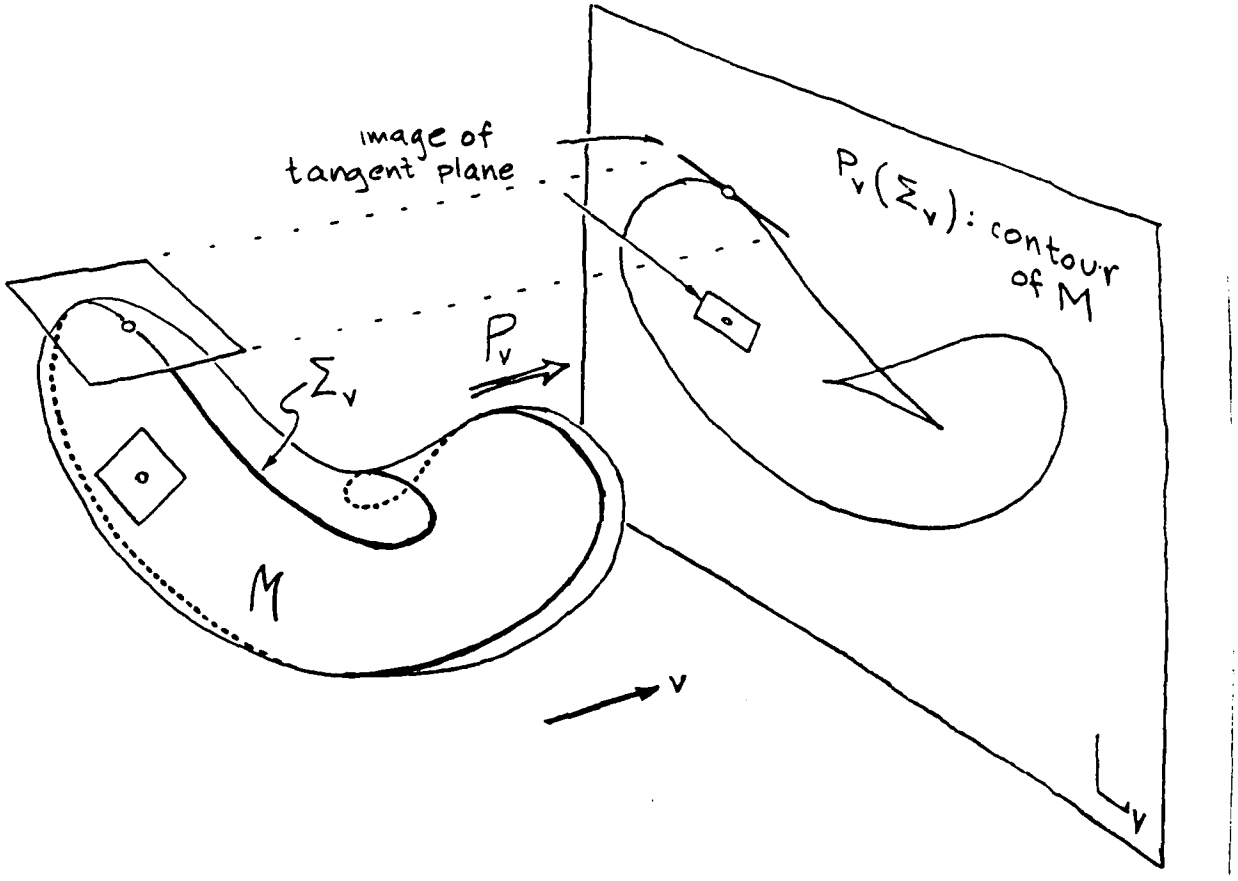


Figure 18: A view of the surface M in the direction v. See text for details.

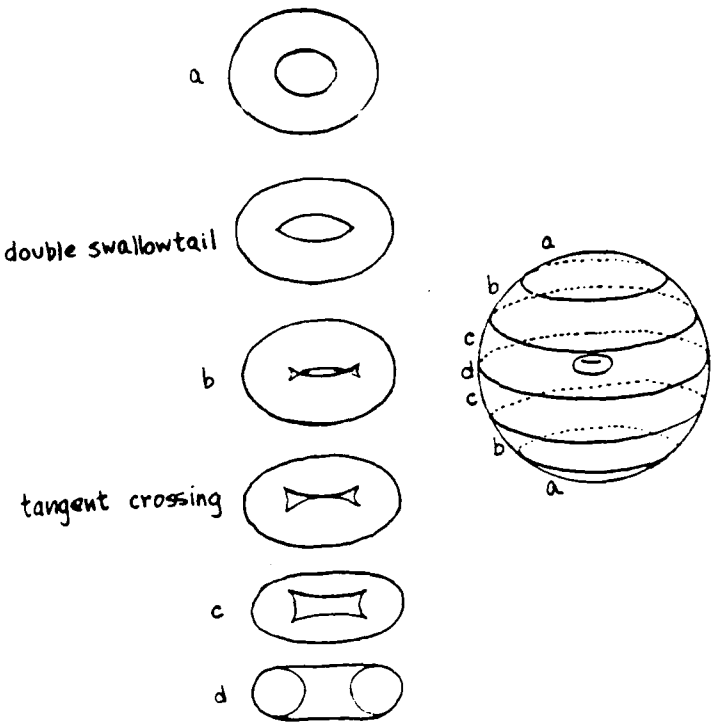


Figure 19: The views of a torus, and their relations to the viewing sphere of a torus.

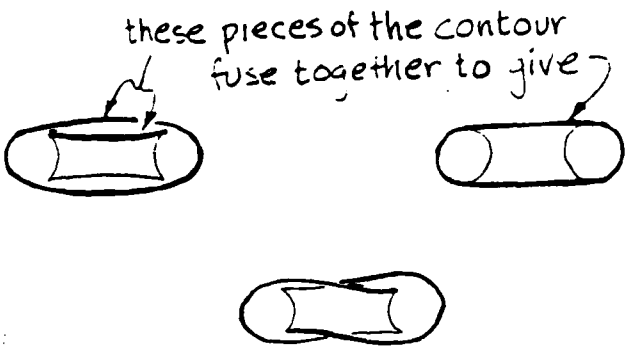


Figure 20: The non-generic torus (upper illustration) is twisted to make it generic (lower illustration).

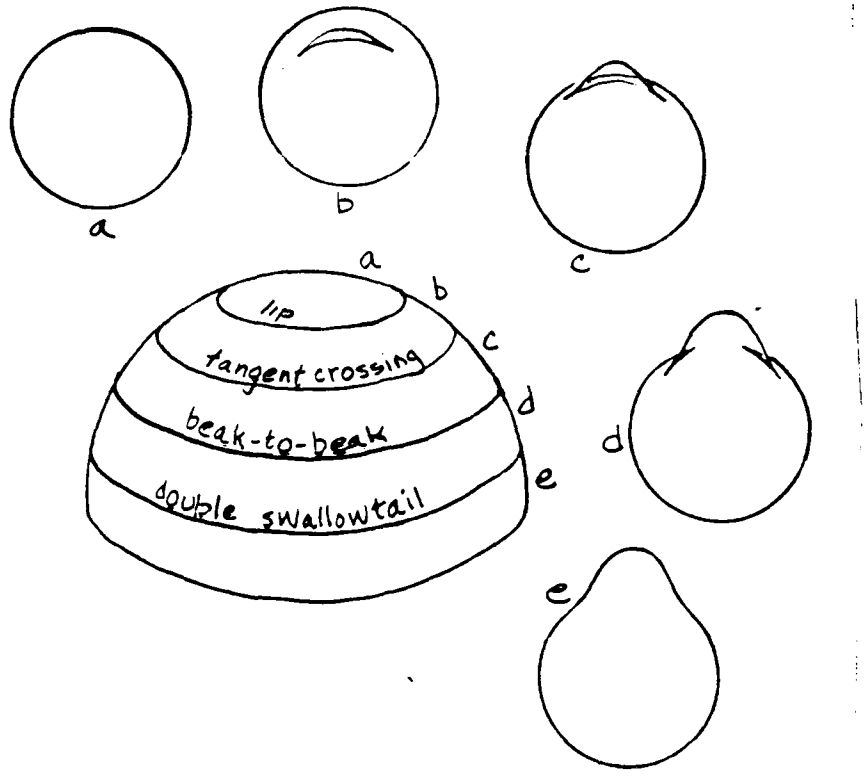


Figure 21: The viewing data for a sphere with a bump.

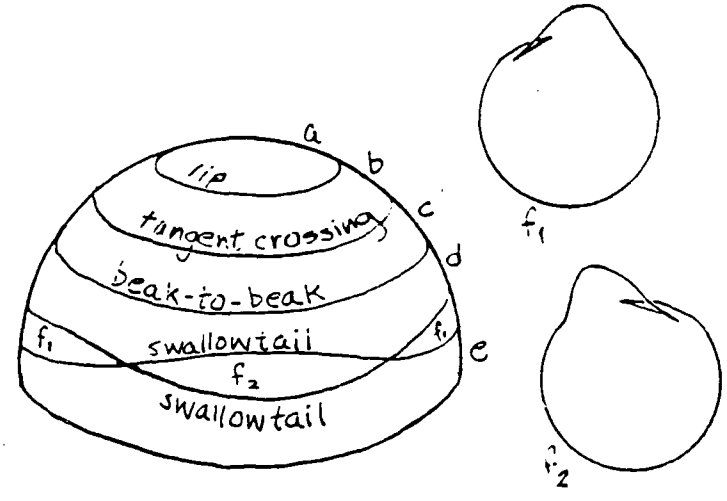


Figure 22: The modifications to the viewing data for a sphere with a bump when the bump is made generic.

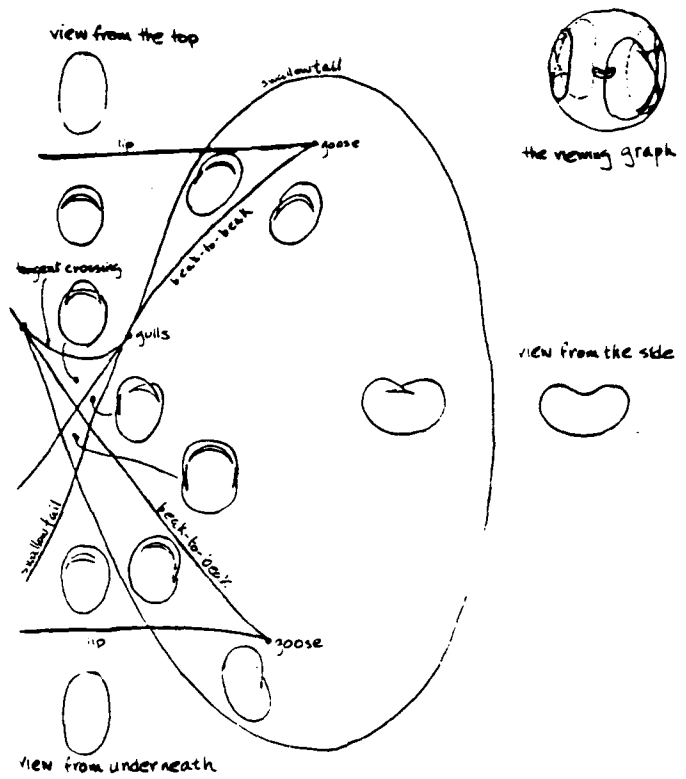


Figure 23: The viewing data for a furrow on a convex body. The viewing sphere, in the upper right hand corner, shows the complete viewing graph. The surface itself—which looks like a kidney bean—is shown at the center of the sphere. The remainder of the illustration is an enlargement of one quarter of the viewing graph. Within each open region of this graph is found the view of the kidney bean which is seen from this viewpoint. Every generic view of the kidney bean is shown.

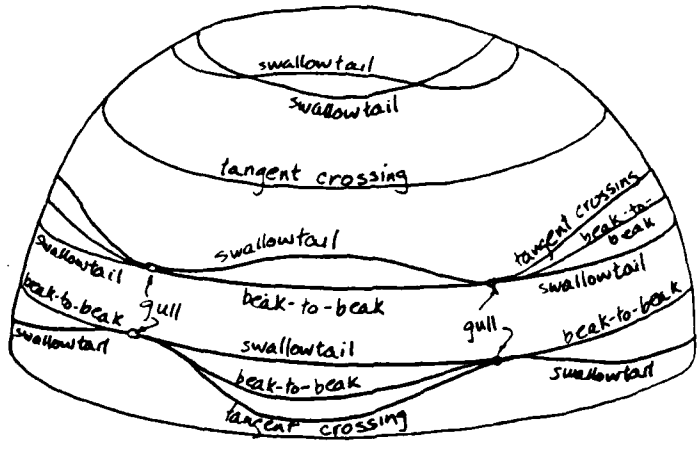


Figure 24: The viewing graph for a generic torus.

4. Reconstruction of surfaces from multiple views

This section presents an algorithm for computing a depth map for a smooth surface from a sequence of images taken from known camera positions. It is assumed that the profiles of an object are extracted from each image.

We can tell a lot about the shape of an object from a single profile, and with multiple views we can often determine the shape uniquely. In this paper we analyze this reconstruction process mathematically and derive an algorithm to produce a depth map. For some applications it may not necessary to produce a depth map at all (for example in recognition problems). The Gauss and mean curvatures may be sufficient by themselves to solve this problem, and in any case it will be useful to decompose surfaces into patches according to whether they are convex, concave, hyperbolic, parabolic, or planar (Besl and Jain [BES86], Brady et al. [BRA84], Ferrie and Levine [FER85]). This gives a method for describing surfaces of objects and a basis for matching with standard surfaces such as spheres, planes, and cylinders. Koenderink and van Doorn [KOE84] have derived a formula which could be used to compute the Gauss curvature from a sequence of intensity images without depth. We have extended these results to produce formulae for the mean curvature, principal curvatures, and principal directions.

The use of profiles for the recognition and description of surfaces has been explored by many people (Koenderink and van Doorn [KOE84], Callahan and Weiss [CAL85], Hoffman and Richards[HOF83]), but most investigations have not combined information from multiple views. In general, there is no way to identify a point on one profile with a corresponding point on a profile from a different view since for smooth surfaces they will not have any points in common. In fact, most stereo algorithms which are based on correspondence find the most similar point and assume it is the same. However, if the camera motion is known, then there is a method to identify points on two different profiles. In our work, we have restricted the camera to planar motion, so that planes parallel to the plane of motion induce a correspondence between the profiles. However, it is possible for the profile to change qualitatively between views, and in order to understand this, we have analyzed the analogous problem for a curve in the plane. These transitions create ambiguities in the reconstruction process. The criterion used to resolve this ambiguity is that the most likely solution is the one which minimizes the change in depth between adjacent views.

A mathematical approach to the reconstruction o surfaces is based on the fact that a smooth surface without inflection points is the envelope of all of its tangent planes. However, there are two problems with this: how to compute the envelope of a family of planes and how to handle inflection points. With the assumption of planar camera motion, we have been able to reduce the envelope of planes problem to that of computing the envelope of a family of lines in a plane, which we were able to solve. The problem of inflection points requires interpolation of the surface. We have a simple approach to this which would work in most cases, and we plan to extend this to polyhedral surfaces where every points is either an inflection point or a crease point (i.e. non-smooth).

In Section 4.1, we give the mathematical framework for discussing profiles. Then for simplicity in Section 4.2, we take the case of reconstructing a plane curve. In Section 4.4, this is generalized to surfaces. In Section 4.6, we present the experimental results using simulated data.

4.1 The profiles of a surface

Let u be a unit vector in 3-space $\mathbf{R}^3$ and let M be a smooth surface in $\mathbf{R}^3$. We regard u as defining a viewing direction. On the surface M there is a locus of points p for which the tangent plane contains the viewing direction. This locus of points is called the *critical set* corresponding to u. If this curve is projected parallel to u onto the viewing plane which is perpendicular to u and passes through the origin, the image is called the *profile* (Figure 1). For future reference we note the following standard facts about critical sets and profiles:

- The critical set is a smooth curve at p unless p is a parabolic point and u is the asymptotic direction.
- The profile near q in the viewing plane arising from p on the critical set is a smooth curve unless the viewing direction is an asymptotic direction.
- When the critical set is smooth at p, its tangent direction is parallel to the viewing plane if and only if the viewing direction is a principal direction at p.
- When the profile is smooth at q, its tangent at q is always in the tangent plane to the surface at p.

In addition, the Gauss curvature can be computed from the profiles. Consider a point p of the critical set. The viewing direction at p together with the normal to the surface there determine a plane, whose intersection with the surface is a smooth curve. The curvature of this curve at p on the critical set is called the radial curvature. The projection of p onto q lies on the profile, and the curvature of the profile at q is called the transverse curvature. Koenderink and van Doorn [KOE84] showed that the Gauss curvature is the product of the transverse curvature and the radial curvature. Brady et al. [BRA84] has also given an independent proof of this fact. The curvature of the profile can be computed directly from an image, and the radial curvature could be computed from a sequence of images.

4.2 Reconstructing plane curves

Consider a smooth plane curve C, which we usually take to be closed, so that it is given by a parametrization $\gamma(t) = (X(t), Y(t))$ such that the derivative is never zero, i.e. $X'(t)$ and $Y'(t)$ are never zero for the same t. As shown in Figure 26, u is the *viewing direction.* The *viewing line* is the line perpendicular to u through the origin. The profile in this context is the set of points on the viewing line for which the tangent line is parallel to u. The position of a profile point can be expressed as $w \cdot (\sin\theta, -cos\theta)$, where w is the signed distance from the origin, O, to the profile point. For example, in Figure 26, there are three points for which $w > 0$ and one for which $w < 0$. Note that if θ produces a value w, then $\theta + \pi$ produces $-w$. Thus we restrict to $0 \leq \theta < \pi$. The collection of all profile points forms a curve in the plane classically called the *pedal curve* of C with respect to O. If θ changes in such a way that a pair of values of w is annihilated and disappear, or a pair is created, then the pedal curve has a singular point (usually a cusp). This happens when u passes through the direction of an inflexional tangent to C, as shown in Figure 27. In a neighborhood of that point, w cannot be a smooth function of θ. In general, there will be parts of C for which w is a function of θ for some range of values of θ. Suppose C_1 is such a subset of C, then the following proposition states that C_1 can be reconstructed from the function $w = w(\theta)$. Any part of the curve which has no inflexions satisfies this property. Another way to say this is that the Gauss map of C_1 has an inverse, i.e. for each direction on the unit circle there is at most one point of C_1 whose normal points in that direction. Such curves possibly with singularities have been studied by Langevin, Levitt, and Rosenberg [ROS] and are called *herissons* (hedgehogs)

Proposition 1 *1. The curve C_1 consists of points*

$$x = w\sin\theta + w'\cos\theta$$

$$y = -w\cos\theta + w'\sin\theta$$

and $w' = dw/d\theta$ is the distance from the profile point to the corresponding point of C_1

2. The radius of curvature of C_1 at the point corresponding to θ is $w + w''$. (Here C_1 is oriented by increasing θ.)

proof: The line through the profile point $w \cdot (\sin\theta, \cos\theta)$ parallel to the viewing direction $u = (\cos\theta, \sin\theta)$ has the equation

$$((x, y) - (w\sin\theta, -w\cos\theta)) \cdot (\sin\theta, -\cos\theta) = 0$$

i.e.

$$x\sin\theta - y\cos\theta = w \tag{1}$$

The curve C_1 is the *envelope* of the lines obtained by eliminating θ between equation (1) and the following equation:

$$x\cos\theta + y\sin\theta = w' \tag{2}$$

This gives the unique solution (x, y) in part 1 of the Proposition. Rewriting this as

$$(x, y) = w(\sin\theta, -\cos\theta) + w'(\cos\theta, \sin\theta)$$

proves that w' is the distance. The formula for the radius of curvature for a curve is given by:

$$\rho = (x'^2 + y'^2)^{3/2}/(x'y'' - x''y')$$

Differentiating the equations in part 1 and substituting into the formula for ρ gives part 2 of the Proposition.

It is not possible for $w + w''$ to be zero while C_1 is smooth: zero radius of curvature is only possible at a cusp of a curve. On the other hand $w + w''$ can tend to infinity, making the curvature tend to zero (which is an inflexion). But at an inflexion w is no longer a function of θ so it is not in C_1.
Examples of these two cases are $w = \theta^3$ and $w^2 = \theta^3$ respecively (see Figure 28). For the former, $x = 3\theta^2$+higher order terms, and $y = 2\theta^3$+ higher order terms. Thus C_1 has a cusp and the pedal curve has an inflexion at $\theta = 0$. For the latter, $x = \pm\frac{3}{2}\theta^{1/2}$+ higher order terms. and $y = \pm\frac{1}{2}\theta^{3/2}$+ higher order terms. Thus, C_1 has an inflexion and the pedal curve has a cusp at $\theta = 0$.
In practice if we start with a curve C, and measure its profile data, i.e. for each viewing direction $u = (\cos\theta, \sin\theta)$ for some range of values for θ we obtain the various values of w for the profile points. Some values of θ will give more values of w than others, unless C has no inflexions, in which case each value of θ will have two values of w. Starting at some value $\theta = \theta_0$, we choose one of the corresponding values to be w_0 and increase θ following w as a function of θ until an inflexion is encountered. This is detected by a change in the number of w-values. In fact θ_0 can be chosen so that it immediately follows after an inflexion. It turns out that one only needs to consider the case when the number of w-values decreases by two. The parts of the curve C lying between inflexions can be reconstructed using part 1 of the Proposition. The computational problem to be solved is this: having chosen w_0 for $\theta = \theta_0$, what value corresponds to $\theta_0 + \delta\theta$? It is not sufficient to simply choose the closest value of w. Since it is possible for the pedal curve to have crossings (See Figure 29), there is the danger of starting on one branch and switching to the other. Note that we are approximating the function $w = f(\theta)$ at discrete points, and we have assumed that this function is continuous since C is smooth. In addition, since w' is the distance from the viewing line to the point of tangency on the curve, we also want w' to be continuous. This can be viewed as a constraint on choosing successive values of w such that w'' is minimized.

4.3 Perspective projection of curves

There is no essential difference in the mathematics of reconstructing curves from profiles obtained from perspective projection. We derive the formulas here for completeness. Assume that the curve C is contained in a circle of radius r. From each point A on the circle we define the viewing line to be the line parallel to the tangent to the circle at A and a distance d away from it. Each tangent line to C through A will intersect the viewing line at a profile point. (See Figure 30) The profile points are identified by their distance w in a counterclockwise direction from the origin of the viewing line, which is the closest point to A. The values of w for θ and $\theta + \pi$ are no longer directly related.

Proposition 2 *For regions of the curve in which w is a function of θ,*

$$x = \frac{rwd\sin\theta + rw^2\cos\theta + rw'd\cos\theta}{d^2 + w^2 + dw'}$$

$$y = \frac{rwd\cos\theta + rw^2\sin\theta + rw'd\sin\theta}{d^2 + w^2 + dw'}$$

Thus in this case also the parts of C between inflexions can be recovered from a knowledge of the function w. However, in this case the formula for the curvature is rather more complicated.

4.4 Surfaces

We would like to reconstruct a surface from its profiles in a way analogous to the method used for curves in the previous section. The situation for surfaces differs in two respects: each profile is a curve and there is a two-parameter family of viewing directions which as unit vectors are points on the sphere $\mathbf{S}^2$. We seek to find all the tangent planes to M, and for this purpose, it is sufficient (and necessary) to know the profiles for a "great circle" of viewing directions, i.e. all directions which lie in a plane. Consider the tangent plane to M at p. The unit tangent vectors in this plane form a great circle on $\mathbf{S}^2$, and this circle intersects the great circle of viewing directions. Hence some tangent line to M at p is in the viewing direction u, so that p contributes a point q to the profile for the viewing direction u. Of course, for opaque surfaces the situation is complicated by occlusion, and it is possible that for some points, none of the singular sets would be visible.

How do we find the tangent plane to M at p? One line in that plane is, of course, the line through q parallel to u. Another line is the *tangent line to the profile* at q (see Figure 31). If the profile is singular at q then the tangent line is interpreted as a limit of tangent lines at nearby smooth points of the profile. If the profile has a crossing at q, then both branches will contribute a tangent plane to M but at different points p of M ([KOE84], [KER81], [CAL85]).

We shall now reconstruct M from the profiles corresponding to a circle of viewing directions (except where the tangent plane is parallel to the plane of the circle of viewing directions). The calculation which follows is local in that it assumes a parametrization of profiles near each point.

Consider a family of viewing directions, $u = (0, \cos\theta, \sin\theta)$ and corresponding viewing planes $y\cos\theta + z\sin\theta = 0$. Each viewing plane will contain a profile of M; we use orthonormal coordinates (x, w) in the viewing plane, where the w-axis is in the direction $(0, \sin\theta, \cos\theta)$ and the x-axis is the same for all of the planes. A point (x, w) in a viewing plane then is the point

$$x(1,0,0) + w(0, \sin\theta, -\cos\theta) = (x, w\sin\theta, -w\cos\theta) \tag{3}$$

in (x,y,z) space. In practice it is the numbers (x, w) which will be measured from an image.

In general, the profile will be be a curve with isolated cusps, and we present the theory for reconstructing the surface from smooth points of profiles. The case of those cusp points is still under investigation. In practice, this should not be a problem since one can compute the surface at points in a neighborhood of a cusp where the profile is smooth.

Assume that the profiles have the form $w = w(x, \theta)$ over a range of values of θ and x, where $w(x, \theta)$ is a smooth function. Thus we assume that over some range of values of x and θ the profiles are smooth and not tangent to the w-axis. So starting with a point on a given profile of M, we choose an axis of rotation which is not parallel to the normal to the profile at that point. It is intuitively reasonable that if the viewing direction remains in the tangent plane, no additional information will be obtained.

Now consider a fixed x and θ, i.e. a fixed point on a particular profile curve. The tangent plane to M determined by this x and θ passes through $(x, \sin\theta, -w\cos\theta)$ and contains the directions:

$$(0, \cos\theta, \sin\theta) \text{ (the viewing direction)}$$
$$(1, w_x \sin\theta, -w_x \cos\theta) \text{ (tangent to the profile)}$$

where $w_x = \partial w/\partial x$. The equation of the plane is therefore

$$w_x X - \sin\theta Y + \cos\theta Z = x w_x - w \tag{4}$$

where we temporarily use (X,Y,Z) as current coordinates in $\mathbf{R}^3$ to avoid the double use of x.

Remark: If the profiles are parametrized as $x = x(t,\theta)$, $w = w(t,\theta)$ for some parameter t, then the tangent plane is

$$w_t X - x_t \sin\theta Y + x_t \cos\theta Z = x w_t - x_t w \tag{5}$$

The surface M is the *envelope* of all the tangent planes described by (4), that is we obtain the point (X, Y, Z) of M corresponding to (x,θ) by eliminating x and θ between (4) and its derivatives with respect to x and θ, viz.

$$\partial/\partial x : \quad w_{xx} X = x w_{xx} \tag{6}$$

$$\partial/\partial\theta : \quad w_{x\theta} X - \cos\theta Y - \sin\theta Z = x w_{x\theta} - w_\theta \tag{7}$$

This amounts to finding the intersection of three tangent planes given by (x,θ), $(x+\delta x, \theta)$, and $(x, \theta+\delta\theta)$. The intersection of these three planes will approach the point on M in the limit as δx and $\delta\theta$ go to zero. This runs into problems when one or the other direction produces a stationary tangent plane. According to equation (6), $X = x$ (the line through a profile point in a viewing direction is always in a plane X = a constant), but (6) also indicates that if the profile has an inflexion ($w_{xx} = 0$), then there will be problems distinguishing one tangent plane to M from the "next". This is similar to the case for curves in which the envelope of tangent lines to a plane curve with an inflexion contains the whole inflexional tangent line. In fact (4), (6), and (7) determine a unique point (X,Y,Z) if and only if $w_{xx} \neq 0$, namely the point

$$f(x,\theta) = (x, w\sin\theta + w_\theta \cos\theta, -w\cos\theta + w_\theta \sin\theta) \in M \tag{8}$$

Note that w_θ is the distance from a profile point to the corresponding point on M.

Remark In the general case where w is not always a function of x, we can parametrize the family of profile curves as follows:

$$x = x(t,\theta)$$
$$w = w(t,\theta)$$

In this case, except where x_t is zero, equation (8) becomes

$$f(t,\theta) = (x, w\sin\theta, -w\cos\theta) + \left(\frac{-x_\theta w_t + x_t w_\theta}{x_t}\right)(0, \cos\theta, \sin\theta) \qquad (9)$$

and $(-x_\theta w_t + x_t w_\theta)/x_t$ is the distance from a profile point to a point on M.
The formula (8) tells us how to reconstruct M from its profile data, as long as w can be expressed as a function of x and θ. The condition for f to give a smooth piece of surface (i.e. the condition that the differential of f has rank 2) is $w + w_{\theta\theta} \neq 0$; note that this also arose in the case of curves above.
On M at any point $f(x,\theta)$, we have coordinate directions corresponding to:

$$\partial f/\partial x = f_x = (1, w_x\sin\theta + w_{x\theta}\cos\theta, -w_x\cos\theta + w_{x\theta}\sin\theta)$$

where f_x is in the direction of the *critical set* and f_θ is in the *viewing direction* so they are not in general orthogonal. Nevertheless, they are *conjugate* with respect to the second fundamental form (see Figure 32). Note that f_x and f_θ will not coincide at a smooth point of the profile. Geometrically, this means that with respect to the ellipse determined by this quadratic form, each direction is tangent to the ellipse at the point of intersection by the axis determined by the other. Algebraically, the matrix associated with the second fundamental form is diagonal with respect to the basis of these two directions and (using $n = (-w_x, \sin\theta, -\cos\theta)/(1+w_x^2)^{1/2}$ as the unit normal to M) can be written as:

$$\begin{pmatrix} \dfrac{w_{xx}}{(1+w_x^2)^{1/2}} & 0 \\ 0 & -\dfrac{w + w_{\theta\theta}}{(1+w_x^2)^{1/2}} \end{pmatrix}$$

Thus, it is possible to derive simple formulae the Gaussian and mean curvatures of M in terms of the profile data w. As noted above, the consequent formula for the Gaussian curvature as the product of the radial curvature, κ_c and the transverse curvature, κ_θ, was known,but the mean curvature cannot be expressed in terms of only these two.

4.5 Relationship between sectional and surface curvatures

There are three curvatures which enter into the equations for the surface curvatures.
κ_c is the curvature of the profile or the radial curvature. Its formula is

$$\kappa_c = w_{xx}/(1+w_x^2)^{3/2}$$

κ_x is the sectional curvature of M in the f_x direction (the direction of the tangent to the critical set). We find (see below) that

$$\kappa_x = w_{xx}/[(1+w_x^2)^{1/2}(1+w_x^2+w_{x\theta}^2)]$$

κ_θ is the sectional curvature of M in the f_θ direction, which is the viewing direction u when $w + w_{\theta\theta} > 0$ and $-u$ when $w + w_{\theta\theta} < 0$. This is also known as the transverse curvature. We find (see below) that

$$\kappa_\theta = -1/[(1 + w_x^2)^{1/2}(w + w_{\theta\theta})]$$

Proposition 3 *1. The Gauss curvature K and the mean curvature H of M at $f(x, \theta)$ are given by*

$$K = -w_{xx}/[(1 + w_x^2)^2(w + w_{\theta\theta})] = \kappa_c\kappa_\theta$$

$$H = \frac{w_{xx}(w + w_{\theta\theta}) - 1 - w_{x\theta}^2}{2(w + w_{\theta\theta})} = \frac{1}{2}\kappa_c\left(1 + \frac{\kappa_\theta}{\kappa_x}\right)$$

2. $w_{x\theta} = 0$ if and only if the viewing direction is a principal direction on M at the corresponding point. In this case f_x and f_θ are the principal directions, $\kappa_c = \kappa_x$, and $H = \frac{1}{2}(\kappa_c + \kappa_\theta)$.

proof: The computation of the Gauss and mean curvatures from a local parametrization f of M can be found in O'Neill [ONE66]. We can obtain f_x and f_θ from equation (8) and compute the first fundamental form as

$$\begin{pmatrix} 1 + w_x^2 + w_{x\theta}^2 & w_{x\theta}(w + w_{\theta\theta}) \\ w_{x\theta}(w + w_{\theta\theta}) & (w + w_{\theta\theta}) \end{pmatrix}$$

This together with the second fundamental form given above allows us to compute the surface curvatures. Thus K and H can be expressed in terms of the curvatures $\kappa_c, \kappa_x, \kappa_\theta$: The shape operator, which is the derivative of the Gauss mapping, referred to the basis f_x, f_θ is:

$$\frac{1}{(1 + w_x^2)(w + w_{\theta\theta})}\begin{pmatrix} w_{xx}(w + w_{\theta\theta}) & -w_{xx}w_{x\theta} \\ w_{x\theta}(w + w_{\theta\theta}) & -(1 + w_x^2 + w_{x\theta}^2) \end{pmatrix}$$

The principal directions are the eigenvectors of this matrix. One can see that, given the assumption that $w + w_{\theta\theta}$ is nonzero, the matrix is diagonal if and only if $w_{x\theta} = 0$. The principal curvatures are the eigenvalues, and the Gauss curvature is the product of the eigenvalues and the mean curvature is their sum. Thus, when the matrix is diagonal, f_x and f_θ are the principal directions and they are orthogonal.

When the profile has a cusp, the curvatures K and H will be the limits of the values of the formulae at corresponding smooth points near it. However, K for example cannot be expressed as $\kappa_c\kappa_\theta$, since κ_c is infinite and κ_θ is zero. Finding a formula to replace this is a goal of current investigation.

4.6 Experimental results

In order to determine the potential accuracy of the algorithm for reconstructing curves, several experiments were performed. Synthetic data was used to generate profile points for the various values of θ. Figure 4.7 shows a picture of a curve with two inflexions and the pieces of the curve which are reconstructed from the data. The reconstruction process stops at the inflexion points producing breaks in the curve. In addition, there is a break in the curve where the reconstruction process started.

In principle, one could start with the profile data and trace the curve from start to finish, reversing direction at inflexion points, always choosing the value of w which minimizes w''. However, there is no sure test on the values for w which will guarantee that a point is an inflexion point. It is possible to find all of the tangent directions for the inflexion points, which can be called *flex directions*. These are the directions at which the number of tangent lines changes. We use these flex directions to break up the profile data into *heurisson* segments for which w is a function of the viewing direction. The current algorithm constructs arcs corresponding to the *heurisson* segments between inflexion points.

As a practical point, one can choose the place to start drawing the curve to be any w value after a flex direction. Once the pieces of the curve between the inflexions are drawn, they can be linked together with straight lines based on proximity of endpoints (assuming one has started with a closed curve. In theory one would like to minimize the integral of the absolute value of w''. In the current algorithm we only apply this criterion locally to choose the values of w sequentially. It may be necessary to apply standard relaxation techniques when dealing with real data.

The algorithm can be extended easily to surfaces, and we intend to apply this to the problem of model acquisition from physical prototypes.

4.7 Discussion

We have given a procedure for reconstructing surfaces from a sequence of profiles. In addition we have given a formula for mean curvature in terms of the profile data which extends the similar result for the Gauss curvature. This makes it possible to compute both of these curvatures without first computing a dense depth map. We have used this algorithm on synthetic, noise-free data to reconstruct curves with a high degree of accuracy, and we plan to test it on real data.

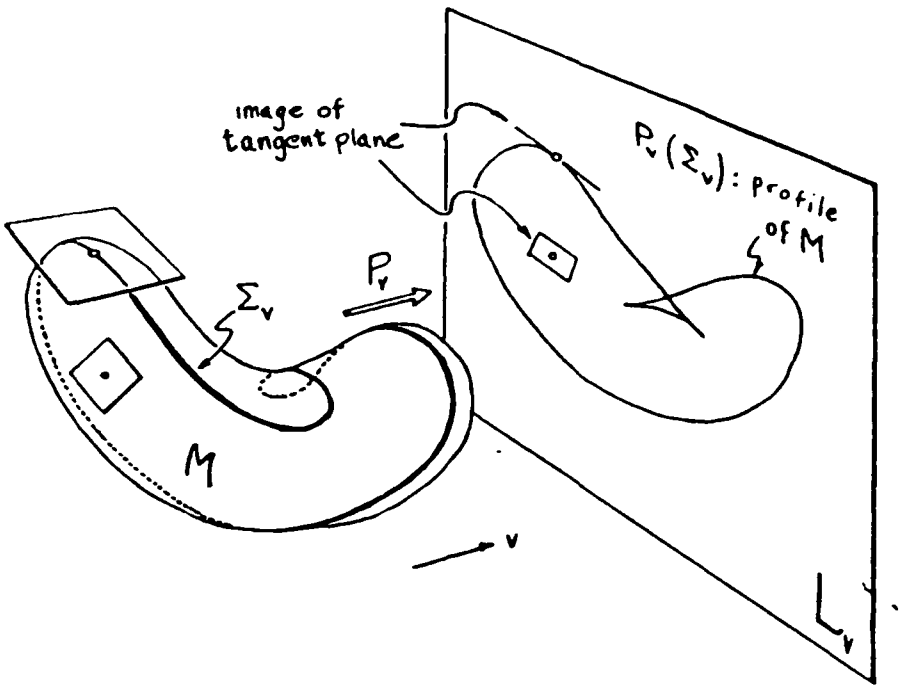


Figure 25: The critical set Σ_M and profile of the projection of M. M is oriented so that there is self occlusion which produces a crossing and two cusps.

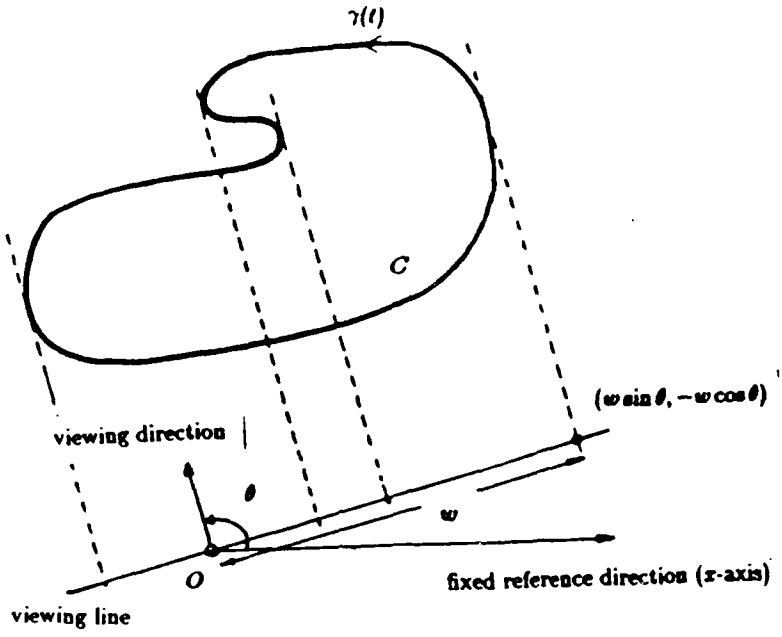


Figure 26: The profile of a plane curve is a set of points on the viewing line

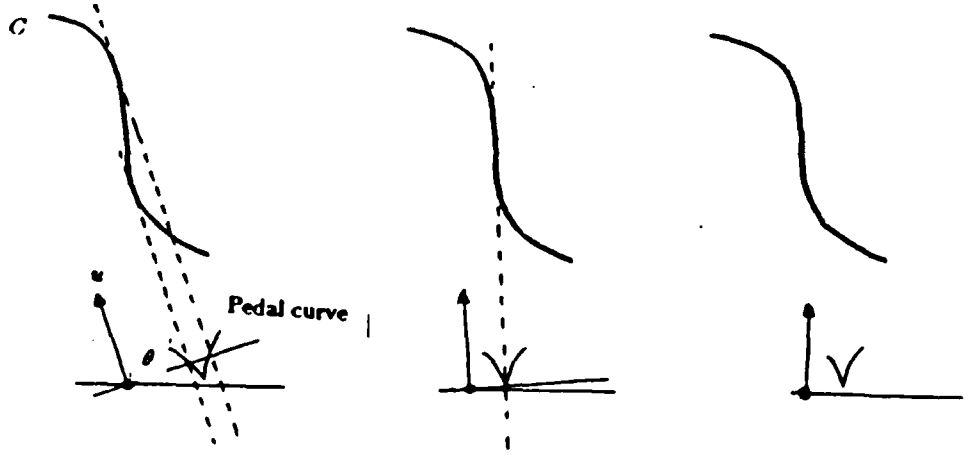


Figure 27: An inflexion in C produces a cusp in the pedal curve

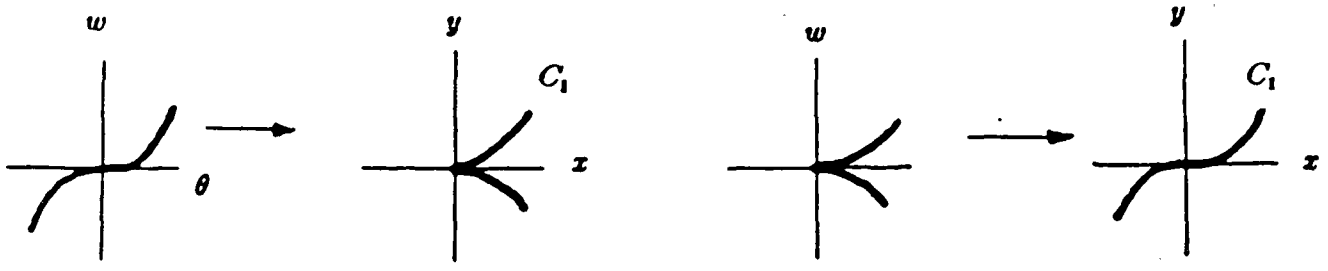


Figure 28: Examples of pedal curves for a cusp and an inflexion

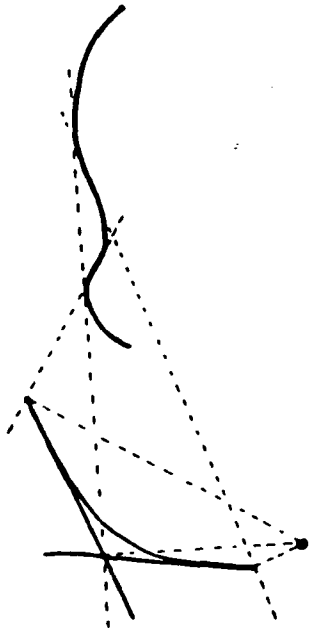


Figure 29: A point where the pedal curve crosses itself

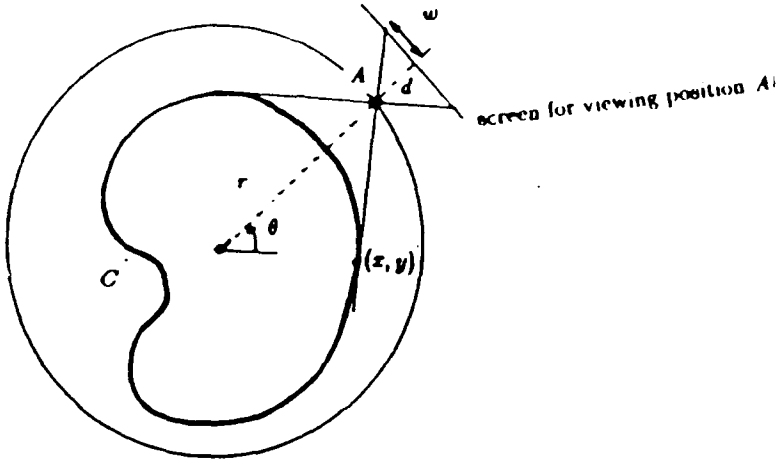


Figure 30: The viewing line for perspective projection

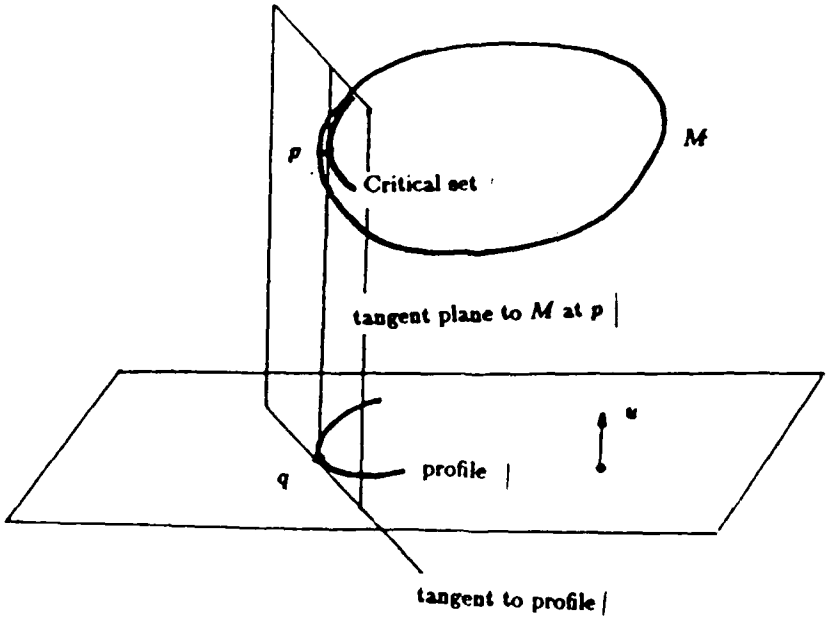


Figure 31: The tangent plane at p contains the viewing direction and a vector parallel to the tangent to the profile curve at q

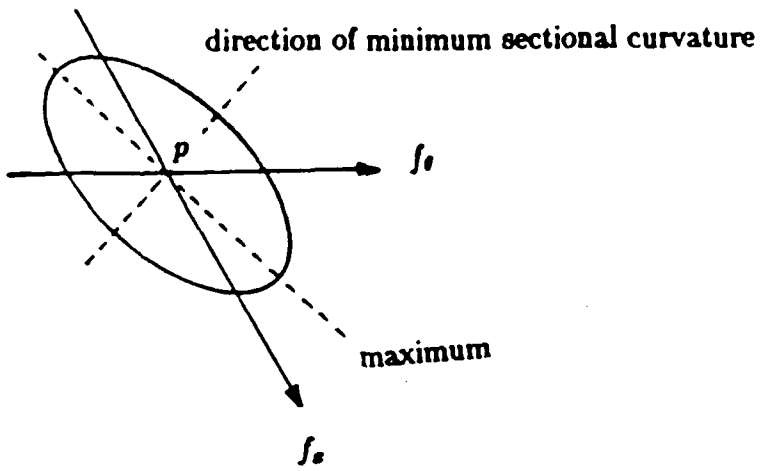


Figure 32: The viewing direction and the tangent to the critical set are conjugate

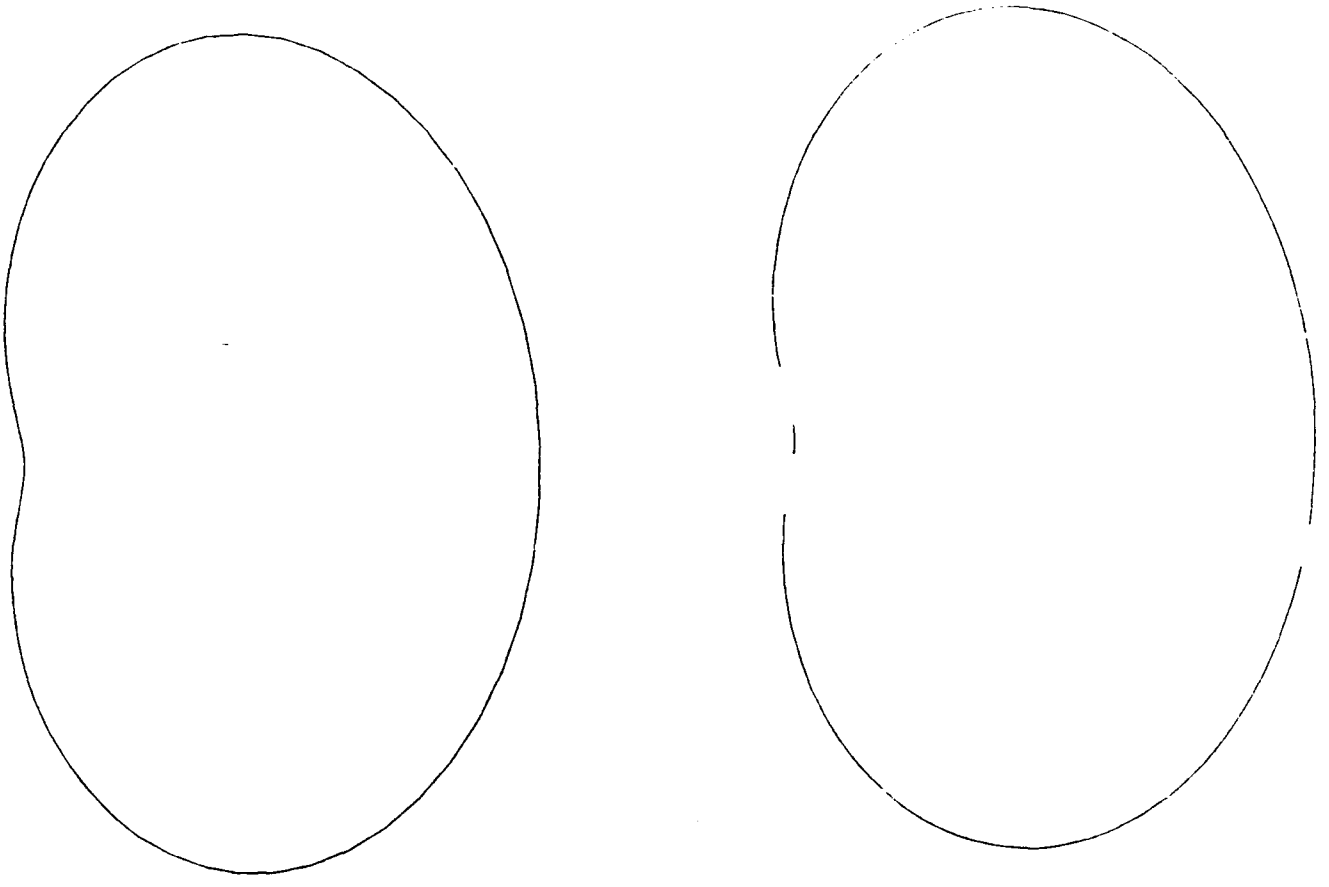


Figure 33: a. A drawing of the limacon from initial data. b. The reconstruction of the limacon.

5. Predictions and the Prediction Hierarchy Compiler

This section describes the design for an object recognition system (see Figure 34), based on a prediction hierarchy compiler, which embodies two major aspects to our approach. The first is that before recognition, in an "off-line" process, we precompile from the 3D model base descriptions of the potential appearance of all the objects from all possible vantage points and thus act as an index into the model data base. (This is actually something of a conceptual simplification: what is actually done will be described later.) These descriptions constitute *predictions* about the 2D image appearance of the objects. By doing this we avoid, at recognition time, the cost of the complicated 3D-to-2D transformations between 3D models and 2D images. The cost of these computations in the general case is one of the chief difficulties with ACRONYM [BRO81b]. In our approach the matching at recognition time is reduced to a 2D-to-2D problem. This precomputation of predictions is possible because, while an object potentially has an infinitude of different exact appearances, only a finite number of these are different in a qualitative sense [CAL85], and for many objects, the number of these qualitatively different views ("generic views") is reasonably small.

Even so, the number of such views over a large model base can still be very great. This leads to the second component of our approach, which is to organize these predictions into a hierarchical structure based on PART-OF (aggregation) and IS-A (specialization) links. This way, the common aspects of the predictions are made explicit and represented just once in the hierarchy. One advantage of this is that the set of predictions can thereby be stored more compactly. But more important, predictions common to many objects need be matched only once during the later recognition step, effecting a considerable economy in matching. Ideally, the hierarchy should act like an efficient discrimination net, permitting rapid indexing to an object-specific prediction. In this way our *prediction hierarchy* provides a single, uniform mechanism for coping both with variation and similarity across objects and across views of a single object.

In reality, these two operations, of prediction and organization, are not run separately, but are intertwined. Predictions are generated and incorporated into the hierarchy incrementally, as needed. This has two related advantages. First is that a potentially unmanageable, unorganized collection of predictions is never actually created; second is that the predictions generated need only be elaborated sufficiently to ensure adequate discrimination between objects (and distinct views). The full, detailed description of the generic views of the objects normally need not be generated; only general predictions of appearance sufficient to adequately discriminate object views one from another are needed. Indeed, in the system as implemented, the concept of *generic view* does not appear at all.

Out of these twin processes of prediction and organization comes a prediction hierarchy, which encapsulates in a directly usable form the knowledge needed to visually recognize objects out of the 3D model base. Notice that, while the construction of the prediction hierarchy may be a computationally expensive process, it need be done only a single time for any given model base. Once the prediction hierarchy is compiled, it can be used over

and over again for recognizing objects out of that model base.

The actual recognition of an object in an image involves matching the hierarchically organized predictions against perceptual features extracted from that image. In the prototype system described here, matching takes place systematically from most-general predictions (shared by many objects) to object-view-specific predictions, thereby ensuring effective indexing from the image data into the model base.

5.1 The Domain

Although our goal is the recognition of objects in natural scenes, for these experiments the domain chosen is polyhedral objects with linear surface markings. Polyhedra form a useful and rich class of objects for recognition experiments, while their 3D modeling and rendering are fairly well understood. In particular, the edges and linear surface markings of polyhedra project into straight-line segments in an image, and good techniques for extracting such straight-line features from images are readily available [BOL86,BUR86]. Note, however, that our overall approach is in no way restricted to polyhedra: in concurrent work at UMass, generating predictions for curved objects is also being investigated [,DOL88, CAL85].

The actual objects used to demonstrate the design are shown in Figure 35 The objects have differences and similarities in various dimensions and part of the problem is to take advantage of both. The similarities in their visual structure, such as occurrences of parallelograms or of certain types of line junctions, must be utilized by the recognizer to make the search for a match efficient. The differences in visual structure, such as height-to-width proportions, must be utilized to discriminate between the objects. Additionally, the variations in visual appearance caused by variations in the camera must be taken into account while doing the structural analysis.

5.2 Predictions

For the construction of predictions our camera model is so-called *normal perspective*, following Brooks [BRO81b]. This is equivalent to orthographic projection with an additional image scaling factor. It provides a simple but useful approximation to true perspective projection so long as the camera is not too close to the object. (Note that the use of normal perspective for predictions does not preclude using true perspective for the pose estimation and verification step.)

A *prediction* is a statement concerning some structural aspect of the image of an object. For example, this may be as simple and general as an assertion that there exists a pair of parallel segments in the projection; or as complex as a description of an image unique to some object. A prediction is represented here as a relational graph; the elements in the graph are projected straight-line segments. The relations associated with arcs in the graph mutually constrain the relative orientations, positions and sizes of pairs of line segments. For example, the relation *parallel* constrains two segments to having the same orientation. The predictions can be viewed as conjunctions of these relations over formal segment arguments.

The objects are expected in any pose, hence all predictions generated are invariant to re-scaling, translation and rotation in the image plane. Because of this and the type of projection assumed, there are only two degrees of camera variation that have a significant effect on the projections being described by the predictions: the two angular components of the viewpoint that sweep out a *viewing sphere* about the object.

Any given prediction may be satisfied by the projection of more than one object in the model base and, in fact, it may be satisfied by the projection of more than one set of line segments on the same object. This is especially true for simple structural statements, such as the parallelogram prediction in Figure 36. For each such occurrence, or *instance*, of a prediction, there may be a wide range of viewpoints for which the set of projected line segments satisfy the prediction. For example, the projections of the 3D line segments a, b, c, and d of the triangular prism model (Figure 36), satisfy the parallelogram prediction over all the viewpoints for which these segments are simultaneously visible (half the viewing sphere).

More specifically, we define a prediction *instance* as a one-to-one mapping from a set of projected 3D model segments to the elements of the prediction's relational graph and the set of viewpoints (the *view region*) on the viewing sphere from which the prediction is valid for these segment-element bindings. For a given model base, there is a set of such instances associated with each prediction. During the compilation of the prediction hierarchy, it is important to keep track of all instances of a prediction and their associated view regions.

A relation between a pair of projected segments used in the predictions is represented by ranges of four relational measures, u, v, α and s (see Figure 37). Let us consider two segments, s_1 and s_2. The vector (u, v) is the position of segment s_2 relative to s_1: it is the displacement of an endpoint of s_2 from an endpoint of s_1 measured along s_1 and normal to s_1, divided by the length of s_1. The angle between them, α, is measured counterclockwise

from s_1 to s_2; and s is the relative scale or length ratio of s_2 over s_1. A relation is defined as an *extent box*, i.e. intervals in u, v, α and s, in order to capture the variation over ranges in viewpoint. For instance, projecting a pair of parallel object segments over all possible viewpoints will generate a set of measurements that have a single value (zero) in the α dimension, but some non-trivial extent in the others. Similarly, the family of projections of a pair of object segments that share an endpoint can be represented by a relation that has the value zero in both position components (u, v).

A relation between projected segments is considered useful if it is valid over a wide range in viewpoints and its extent box is small in volume (for example, consider the two view-invariant relations just mentioned). The latter property is important if the relation is to help in the indexing process; that is, to characterize an object's projection with a specificity sufficient to discriminate the object from a large number of other objects and from chance arrangements of image segments. Although invariant relations are clearly useful [LOW87], alone they are not in general sufficient to fully characterize projections. For instance, proportions are often strong characterizations of object structure, but the length measurement ratios that represent them are often not strictly view invariant. For example, the tall box in Figure 35 has a height to width ratio that is significantly different from the cube over a large range in views and no other property can be used to differentiate them.

Any given prediction in the *prediction hierarchy* has an associated relational graph, but explicitly it is almost always represented as some combination or specialization of other predictions (see Figure 38). A prediction is a *specialization* of another if it can be described by adding new relations or narrowing the extent boxes of existing ones (i.e., tightening the constraints). For example, the rhombus can be described by adding a relation between the bottom and side segments of the more general parallelogram prediction that constrains the ratio of their length to be one. A prediction is a *combination* of other predictions if it can be described as a conjunction of these other predictions. Predictions may be combined in various ways, depending on the mappings between the relational graph elements of the part predictions and those of the whole combination. Figure 39 shows an example of how the nature of the combination prediction can change by varying these *part-whole* maps, which are represented by the arguments passed into the part predictions and the ordering of these arguments.

Note that, because of the combination predictions, the "hierarchy" is actually a directed acyclic graph, where the predictions are *nodes*, and the specializations and combinations are *successors* of the nodes they are created from (their *predecessors*). For the prototype system presented in this paper, the hierarchy compiler and matcher have only been developed to use the combination process. A compiler that uses specialization will be presented in forthcoming work [BUR89]. Also, the compiler currently generates combinations that are limited to pairs of parts.

5.3 Prediction Hierarchy Compilation

In Section 5.2 it was shown that a prediction hierarchy is composed of predictions that are combinations or specializations of other predictions. The primitive, or base-level predictions of the hierarchy (those that all the other predictions are derived from) ought to be those that are valid for very large numbers of objects and are essentially invariant to viewpoint; such predictions are like the invariant binary relations discussed above and in [LOW87]. The final or *goal* predictions ought to be valid for only a few objects and over a narrow enough range of viewpoints that pose estimate refinement and verification can be performed effectively.

A useful way to build such a structure is to start with a small set of simple, ubiquitous predictions and then iteratively search for useful combinations or specializations, eventually isolating predictions that characterize the projections of specific objects. This builds the hierarchy in the direction in which the indexing process proceeds through it, generating successors that seem most useful for discriminating the possibilities associated with each current node.

Though the major concern of this work is the efficiency of the recognition process, the pre-recognition compiler design should also avoid combinatorial explosions that, for a fair-sized model base, would keep it running until the next century. By using this iterative construction approach, we limit the combinatorial complexity, and hence processing time, to a manageable level. This is because the system is never performing isomorphism analysis over large graphs: it is always comparing combinations of small numbers of parts.

5.3.1 Criteria for Selecting Successors

In building the prediction hierarchy, it is important to understand which of the possible combinations and specializations of simpler predictions are useful to explicitly represent as nodes in the hierarchy. For maximum efficiency in the search for the exact object and view, the hierarchy should be in the form of a binary search structure. To achieve this, each new node added by the compiler should be satisfied by exactly half of the instances of its predecessors (where instances are the possible segment-element maps of a prediction described in Section 5.2). This would make the new node an ideal test in a binary search process.

This is roughly what the compiler strives to build, though there is an additional consideration that changes the picture somewhat. The *absence* of a match to a prediction is not considered here as evidence for the presence of alternative objects or views. For each alternative subset of possibilities (instances), there should be an associated successor that requires additional positive evidence in the image in order to be satisfied, i.e., some new image part that needs to be present. This is because the absence of a match could be due to noise, instead of the absence of the object; and noise is more likely to prevent matches than to create spurious matches to a non-trivial relational graph description. For example, if the recognizer has reached node 8 in Figure 40 and is trying to decide whether the object projected is a triangular prism or something else, the absence of the triangle match does

not necessarily mean that it is a projection of some other object, but the presence of an additional parallelogram in place of the triangle is strong evidence that it is indeed another object.

Negative evidence is not used for discrimination. Instead, the compiler associates each subset of instances with a distinct combination successor. The number of successors per prediction (i.e., the branching factor of the hierarchy) should be kept low since large branching factors can slow the match search process down. In addition, it is important that every instance of a prediction be associated with *some* successor of this prediction. Whenever there is a match to a non-goal prediction, there should also be a match to at least one of its successors, so that there can be some path, via a sequence of matches, that culminates in a goal prediction match. An instance is *covered* if there is a successor that is also satisfied by the same object, same view and compatible segment-element bindings (given the part-whole maps relating predecessor elements to successor elements).

In the ideal case then, there would be two successors per node, each with positive evidence (such as the triangle or parallelogram above), and together they partition the prediction instances exactly in half. This ideal case cannot always be met: there may not be successors that are satisfied by half of the instances, but only by somewhat smaller or larger fractions; and there may not be a set of successors that partition the instances into disjoint subsets. Nevertheless, at each iteration the compiler should try to select as small a set of successors as possible that still cover all the instances of the predictions generated the iteration before.

5.3.2 Iterative Hierarchy Construction

For the experiment reported here, *parallelism* and *endpoint coincidence* are used as the initial set of primitive predictions since they occur frequently in the objects studied here, they can be rapidly matched and their combinations can be used to discriminate between most of the objects and views. The iterative construction process stops when all of the predictions without successors are either associated with single objects or, if they are satisfied by projections of more than one object, there are no differences between their projections that the compiler is capable of representing.

For each iteration of the combination process, the system isolates useful combinations by (1) finding predictions that often appear together in the same projections, and then (2) selecting a subset of these commonly occurring combinations that approximately satisfy the criteria discussed in Section 4.1

The co-occurrences between predictions are found by noting the amount of overlap in the view regions of their instances. All instances of all predictions are stored in data structures called *visibility maps*. There is a visibility map for each object; the maps are arrays of cells indexed by the two viewpoint parameters, making a discrete sampling of the view sphere about the object. Each cell lists prediction instances visible from the associated viewpoint and object; and with each prediction is a list of cells that contain it. To find frequent co-occurences between some prediction p and other predictions, the

system looks for predictions that appear in the same cells as p and accumulates the total number of cells for each that do.

Besides visual co-occurrence, an additional constraint is added to the pairs of predictions considered for combination. In order for the resulting successor prediction to a be a spatially compact and connected structure (instead of corresponding to widely separated parts of the projection of the object), only pairs that share at least one projected line segment are considered. For example, the two parallelogram parts of the triangular prism projection in Figure 38 share a segment.

After finding all commonly occurring combinations at the current level, the compiler tries to select a small subset that collectively and efficiently cover the instances of the predictions that they are successors of. The successors are iteratively selected in the following way:

1. Select the successor that covers the largest number of previously uncovered instances.

2. Update the records of all the instances just covered by this new successor.

Note that this iterative selection process is an inner loop to the iterative process that is building up the prediction hierarchy level by level. This inner loop works within a level to select a useful set of successors at that level.

5.4 Example Results

The prediction hierarchy compiler was run on the model base of polyhedral objects shown in Figure 35. Specialization was used in only a rudimentary way, to distinguish the tall box from the cube at the very end. All the other predictions were formed by combination.

The resulting hierarchy is shown diagrammatically in Figure 40. As can be seen, it is a quite reasonable representation of visual knowledge about these objects, organized for efficient recognition. Other prediction hierarchies are also possible given the objects modeled. For example, node 6 of Figure 40 could alternatively have been the combination of a triangle with a parallelogram, rather than the U-shaped structure. This might actually be preferred for greater noise insensitivity, since the parallelogram could afford to lose more line segments to noise and still distinguish the triangular prism from the other objects. Since the compiler is a single-pass algorithm, only combining predictions that have already have been generated, it noticed the triangle and U-shape combination before the parallelogram was constructed. It may be possible to add a post-processing step that realizes more intelligent combinations than can be isolated in a single-pass system.

5.5 Matching

This section is a brief summary of the matching system currently under development. The system tries to find matches to the object-specific goal predictions given a prediction

hierarchy and matches to the primitive, base-level predictions. Starting with these readily found base-level matches, the basic idea is to repeatedly attempt to extend existing matches into matches of more complex successor nodes until good matches to goal nodes are achieved.

The following problems are being investigated in the process of designing this system:

1. Missing straight-line segments in the image, or those "not seen" by the matcher because they are mis-measured, can block match extensions that are steps towards a goal node match.

2. For a given match, there may be a large tree of direct and indirect successors of the node matched. The matcher should avoid exhaustively searching this tree for matches to successors.

3. Because predictions are often represented as combinations of predecessors that may themselves be large and complex, the attempt to extend a match to a successor match can involve a costly search for the other predecessor. These searches should be avoided, or methods of reducing their cost should be implemented.

4. Image clutter can generate a lot of possible match extensions, potentially slowing down the search for goal matches.

5. Symmetries in a node's relational graph description can slow the search for matches to its successors. There may be many possible locations to check for the match of the other predecessor. There also can be ambiguities to resolve in the image-successor match bindings, given the matches to the predecessors.

We believe that a system of the following basic design can handle these problems. It is a set of asynchronous matching processes that are distributed spatially about the image. Each process repeatedly attempts to jointly extend *pairs* of existing matches whose image segments lie in a spatial zone associated with the process. The extension operation has the following steps:

- Select a pair of existing matches whose image segments have some non-accidental relationship, such as those discussed in [LOW87], or matches that actually share an image segment. Segment sharing can be considered a certain kind of non-accidental relationship: the identity relation.

- Search the prediction hierarchy for common - and possibly indirect - successors to the pair of nodes matched. This is done using a marker passing and collision detection scheme.

- Attempt to complete a match to the common successor given the pair of predecessor matches and the part-whole maps between their relational graph elements. The completion process may involve a search for particular image segments if the common successor is indirect.

For typical images, a number of the expected straight-line segments are missing or mis-measured. In spite of this incomplete information, the objects are often readily recognizable. For a recognition system to detect objects in such data, it must have some ability to find and accept good partial matches, where the quality of the partial match is some function of the fraction of well-matched segments. Because the absence of segments may prevent a good partial match of a direct successor, but not to an *indirect* successor, the system should be capable of extending a node's match to one of its indirect successors. For example, extending the match of A in Figure 41 to a satisfactory match to B may be impossible; but a match to D, an indirect successor of A, may be good enough in spite of the missing segments.

Each node may have a potentially large tree of successors, and matching to a successor can be involved. Thus the attempt to find matches to direct or indirect successors should not be an exhaustive search of the tree of possible successors. This is where the strategy of attempting to jointly extend a *pair* of existing matches helps: it focuses the search to intersections of the successor trees of the matched nodes. These intersections tend to involve much smaller numbers of successors, and their matches are more promising since *two* predecessors have already been matched. Joint extension of pairs of matches is especially useful when the nodes have symmetries. In these cases, a thorough search for a successor, given a match to only one predecessor, may involve checking many image locations for the match of the other, missing predecessor.

Pursuing joint extensions of pairs of matches does help reduce the amount of effort the matcher spends in the costly search for complex predecessor matches; but, for the system to rely on joint extension, pairs of jointly-extendible matches have to be generated. Also, both members of such a pair should be generated within a reasonable time frame (i.e., there should not be a long delay between their creations). By following a simple depth-first policy, the matcher may often attempt a large number of alternative matches to roughly the same image data, and thus delay the generation of matches over other image data. This is undesirable because predecessors of a node tend to represent parts of the node's relational graph description that are spatially distributed. In other words, jointly-extendible matches may share some image segments, but they tend to occupy different portions of the image. Thus, if matches are being attempted in different parts of the image at roughly the same time and level in the hierarchy, there is a better chance for jointly extendible pairs of matches to occur and hence a more rapid convergence to a goal match. This can be done by explicitly allocating the system resources to matching in distinct spatial zones distributed about the image. This should be done whether or not the hardware is capable of doing these matching operations in parallel.

A fair number of matches to intermediate nodes may be generated in a complex, cluttered image during the pursuit of matches to goal nodes, and thus the number of possible *pairs* of matches to which joint extension can be attempted could be large. The answer is to restrict the joint extensions to pairs that are much more likely to be part of the same object's projection. In complex images, this could dramatically reduce the number of pairs attempted, and these pairs have a much greater chance of being extended. A good heuristic

for doing this is to select pairs of matches whose image segments have what Lowe[LOW87] describes as associations that are not likely occur by accident. For example, the matches of A and C in Figure 41 share an image segment, which almost always occurs when the matches are to projections of the same object.

5.6 Pose Estimation and Verification

In most applications of object recognition it is necessary not only to identify the viewed object, but also to determine its *pose* (position and orientation) with respect to the camera. Matching using the prediction hierarchy may only determine the approximate view of an object. However, it does facilitate exact determination of object pose. The correspondence between the matched prediction and the image features can be related back through the prediction to the original 3D object model, giving a correspondence between the image features and the object model. From this information can be determined the parameters of the 3D-to-2D transformation that best matches the 3D object features to their corresponding 2D image features. What is more, the approximate view associated with the matched prediction can be used as a good starting point for iterative methods of solution for the pose parameters, such as is used in Lowe's SCERPO system [LOW87].

The exact determination of pose also provides verification of the object identification achieved by matching through the prediction hierarchy. This is important because it means that the discriminations made by the prediction hierarchy need not be perfect. It is sufficient that matching through the prediction hierarchy be conservative, that it not rule out any true object identifications. Any objects and views erroneously indexed by the prediction matching can be ruled out by the pose estimation and verification. Of course, the more false indexing ruled out by the prediction matching, the less work there is to be done by the pose estimation and verification, but there is clearly a trade-off involved here. At a certain point it may be more efficient to take a matched general prediction and verify the objects it implies one by one through pose verification, than to further extend the match to an object-specific prediction. One advantage of our approach is that this cut-off point could be determined during the prediction compilation. It is our belief that normally this cut-off point is quite late, that is, the goal nodes of the prediction hierarchy would be specific to an object-view, or at most to a few object-views.

Pose estimation for verification is related to Lowe's use of the *viewpoint consistency constraint*[LOW87], although he uses it only in the context of single-object recognition.

5.7 Summary

We have described an approach to rapid object recognition from large model bases. A structure called a *prediction hierarchy* is used to organize knowledge about the visual appearance of objects in a form directly usable for rapid indexing of the model and general view. We present a prototype implementation of the prediction hierarchy compiler and show its results using a small model base of polyhedral objects. In addition, a recognition

system is proposed and discussed that can efficiently search for matches to the object-specific goal nodes of the hierarchy.

Current work [BUR89] includes further development of the recognition system and a prediction hierarchy compiler capable of specialization. Future directions for this research include a multi-pass prediction hierarchy compiler that is more intelligent in its selection of successor nodes and also the compilation of sophisticated control strategies into the hierarchy.

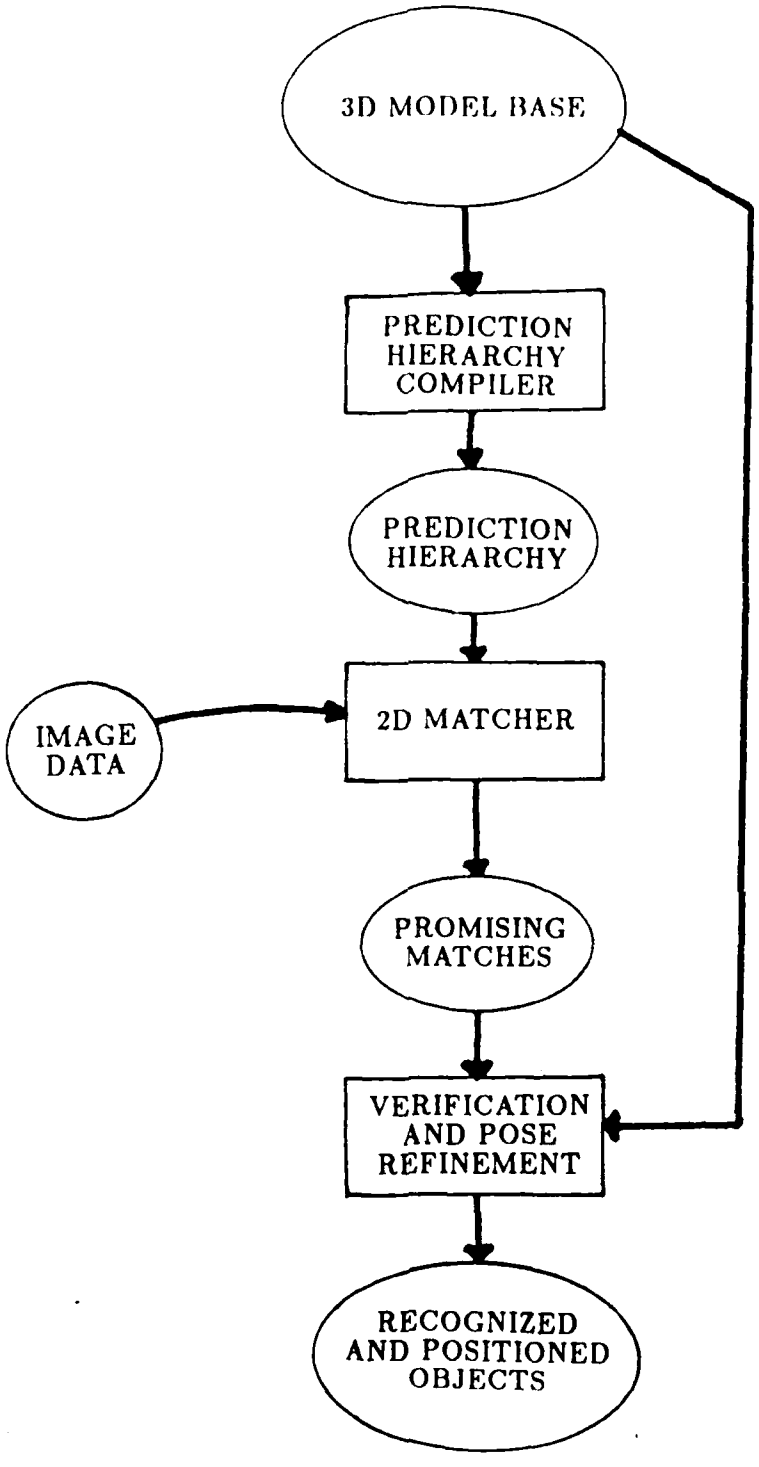


Figure 34: The basic recognition system.

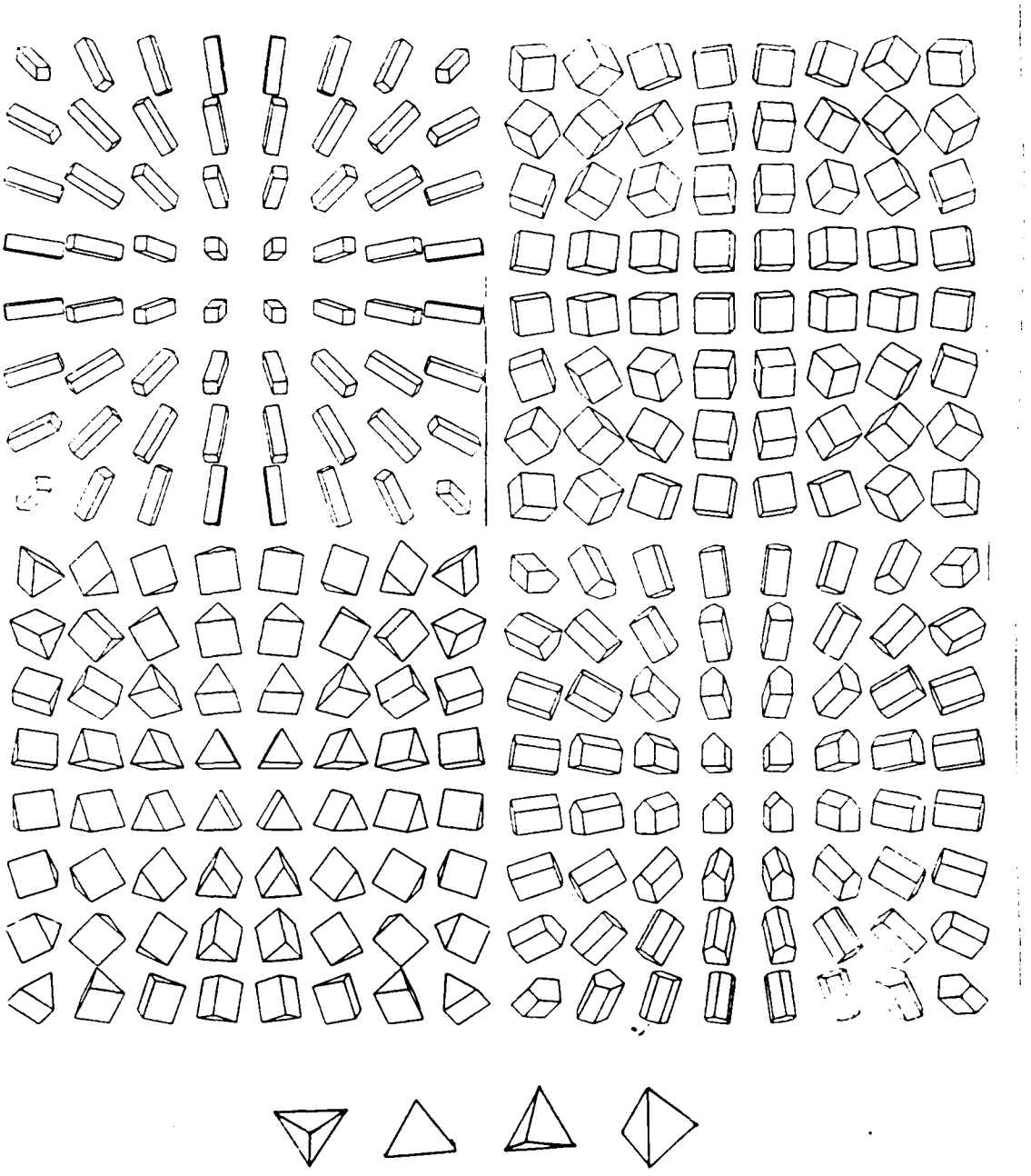
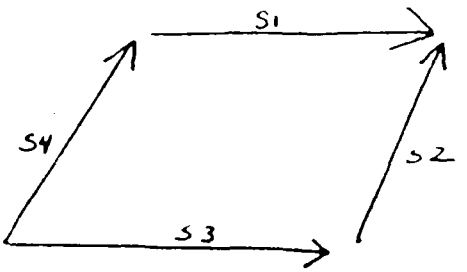
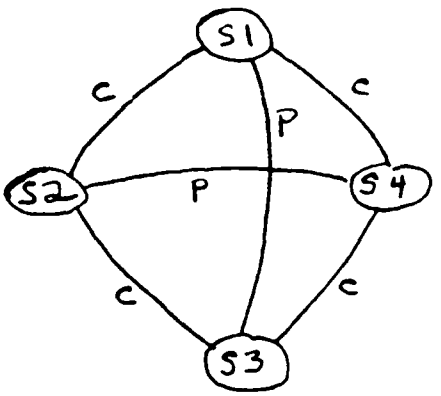


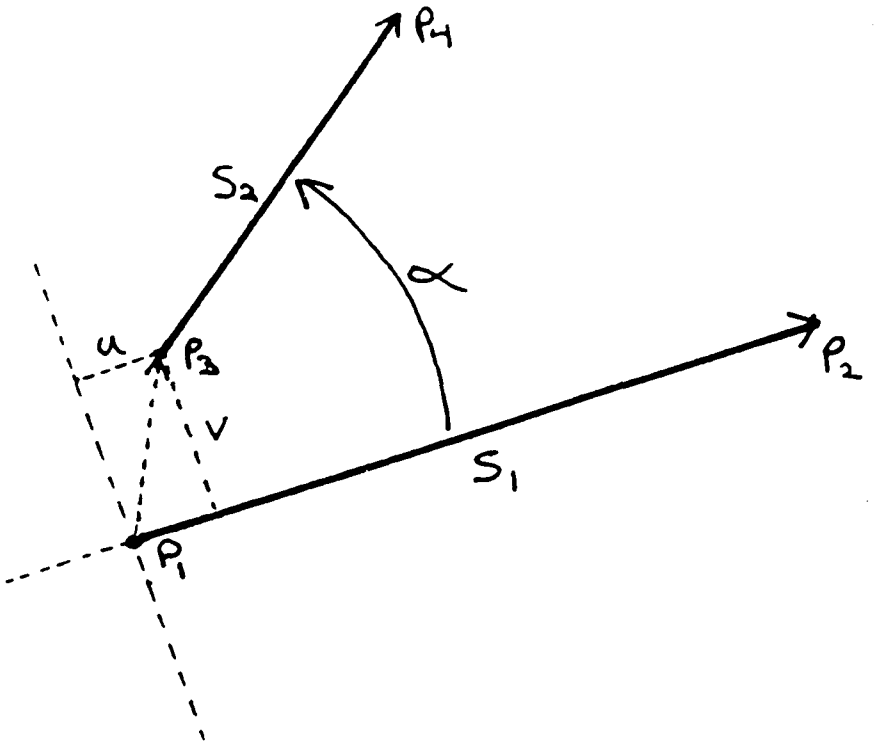
Figure 35: 3D objects used to demonstrate prediction hierarchy






```
(define parallelogram (s1 s2 s3 s4)
  (and (coincident s1-head s2-head)
       (coincident s2-tail s3-head)
       (coincident s1-tail s4-head)
       (coincident s3-tail s4-tail)
       (parallel s1-tail s3-tail)
       (parallel s2-tail s4-tail)
       )
  )
```

Fig. 36: A parallelogram and its relational graph representation. In the internal representation, the segments have arbitrary but fixed endpoint orderings (head/tail). This is to differentiate the parallelogram from other, distinct structures such as a four-way junction (which are otherwise identical).




```
Relative measurements:
----------------------

   u = displacement of p3 from p1 in direction along s1,
         divided by the length of s1
   v = displacement of p3 from p1 in direction normal to s1,
         divided by the length of s1
   alpha = angle between s1 and s2 measured counter-clockwise from s1
   s = ratio of size, length(s2)/length(s1)


Relation as extent box in measurement space:
--------------------------------------------

   R(s1,s2) = (and (min-u ≤ u ≤ max-u)
                   (min-v ≤ v ≤ max-v)
                   (min-alpha ≤ alpha ≤ max-alpha)
                   (min-s ≤ s ≤ max-s)
                   )
```

Figure 37 Representation of image segment relations as an extent box in space of relative position, orientation and size measurements.

```
(define parallelogram (s1 s2 s3 s4)
  (and (coincident s1-head s2-head)
       (coincident s2-tail s3-head)
       (coincident s1-tail s4-head)
       (coincident s3-tail s4-tail)
       (parallel s1-tail s3-tail)
       (parallel s2-tail s4-tail)
       ))
(define triangle (s1 s2 s3)
  (and (coincident s1-head s2-tail)
       (coincident s2-head s3-tail)
       (coincident s3-head s1-tail)
       ))
(define triangular-prism (s1 s2 .. s8)
  (and (parallelogram s1 s2 s3 s4)
       (parallelogram s5 s2 s6 s7)
       (triangle s3 s7 s8)
       ))
(define rhombus (s1 s2 s3 s4)
  (and (parallelogram s1 s2 s3 s4)
       (same-size 1 2)
       ))
```

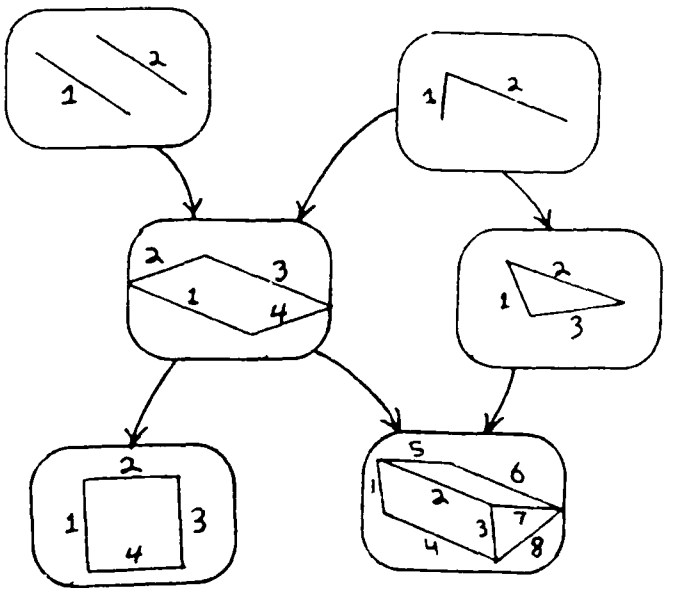
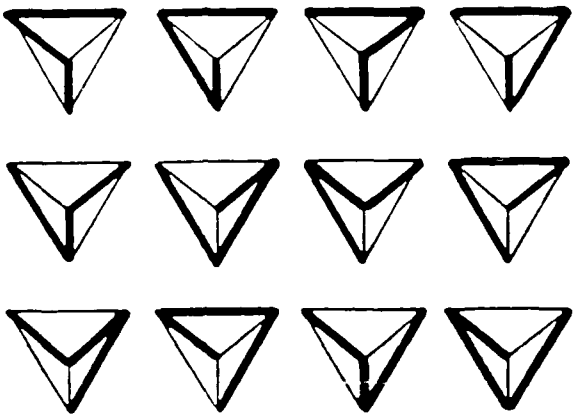
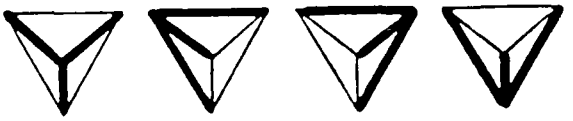


Figure 38 Predictions as specializations or combinations of other predictions.



(a)



(b)

Figure 39: A view of the tetrahedron that contains 12 instances of the *endpoint coincidence* prediction. The instances of two of its derivatives, (a) *snake* and (b) *trihedral junction*, are shown. Both derivatives cover all coincidence instances, but the trihedral junction prediction does so with much fewer instances (4 instead of 12). The snake prediction has three segments connected in a chain that is not closed in a loop; and the trihedral junction has three segments with each pair coincident, using the same endpoints.

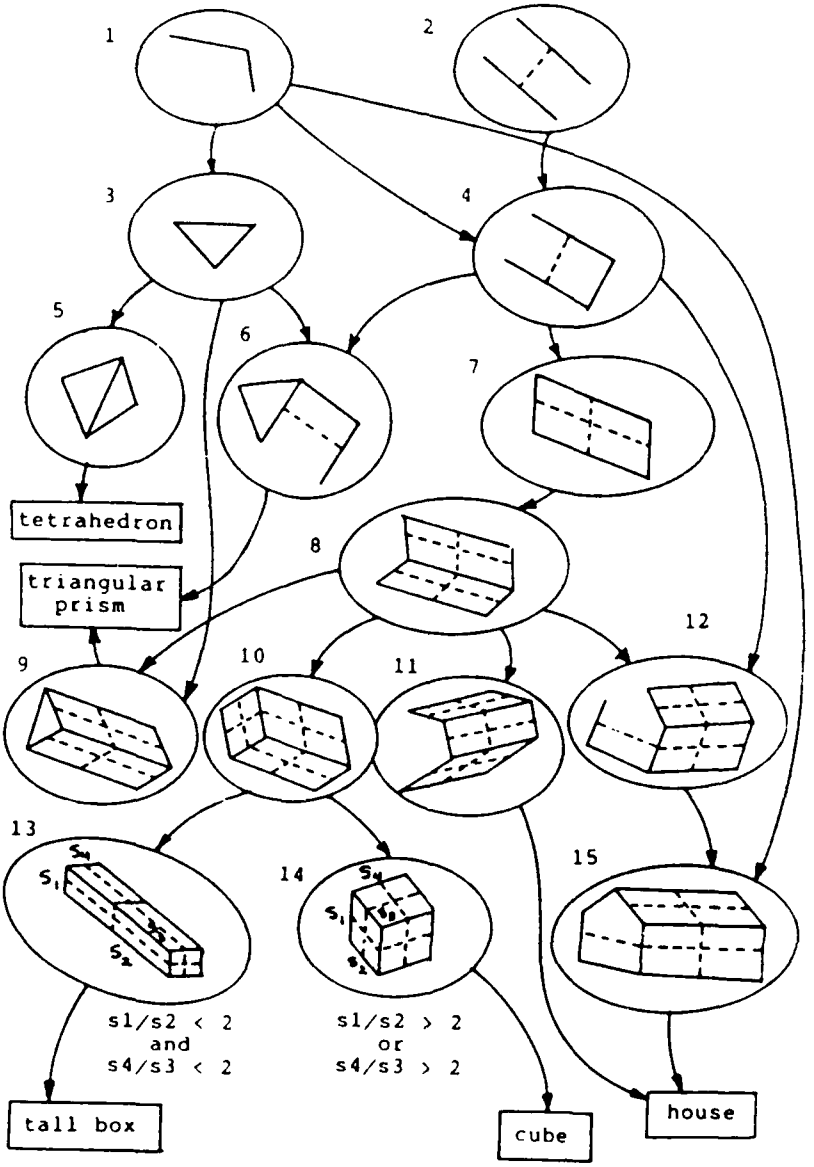


Figure 40 The resulting prediction hierarchy compiled from views of the objects in figure 4. The nodes represent predictions and arrows indicate combination and specialization links. The predictions are represented graphically by segments and dashed lines for parallel relations.

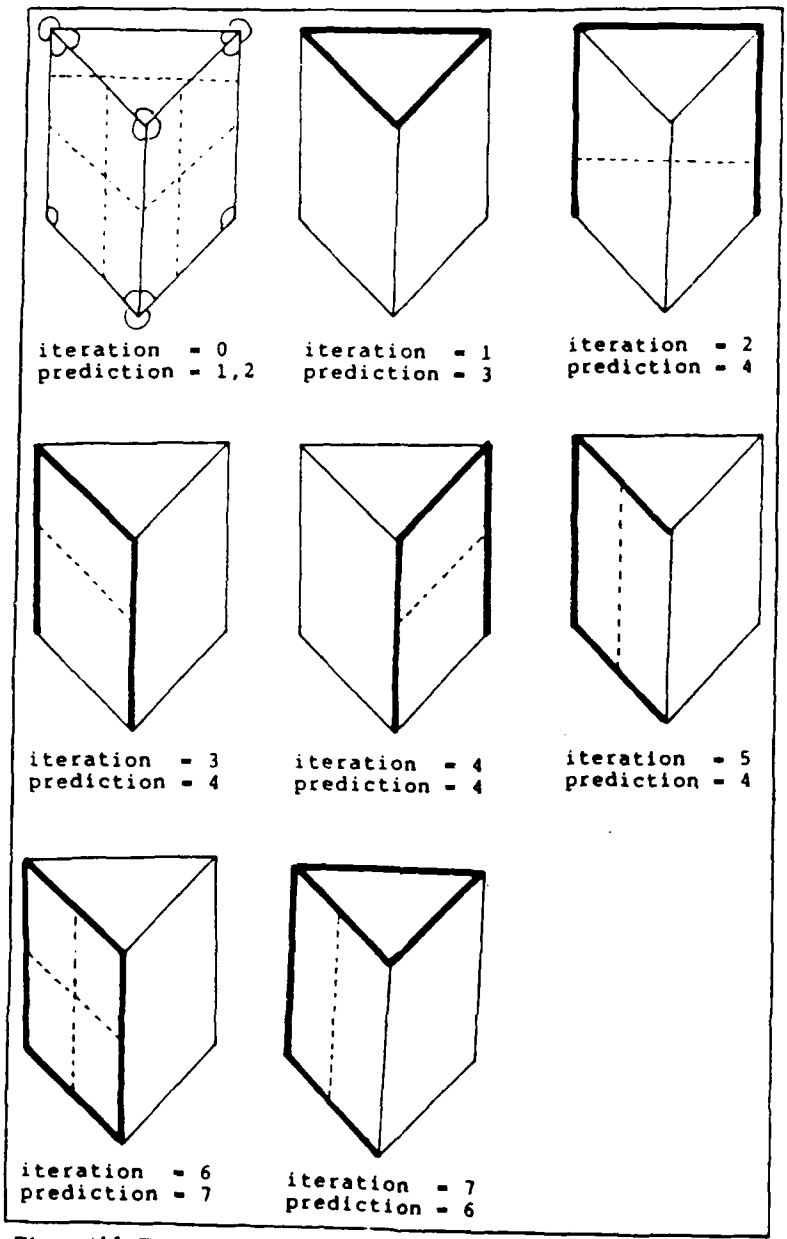


Figure 41. Example run of the matcher using the prediction hierarchy in Figure 12. The matcher is initialized (iteration=0) by finding all instances of the initial predictions (coincidence and parallel). The matcher then iteratively searches for matches between combinations and specializations of already matched predictions and the image. A prediction unique to the triangular prism object was matched to the image at iteration 7.

REFERENCES

[ARN83] V.I. Arnold, "Singularities of systems of rays", *Russian Math. Surveys*, 38:2, pp. 87–176, 1983.

[BAL82] D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, New Jersey, 1982.

[BAL83] D. Ballard and D. Sabbah, "Viewer independent shape recognition", *PAMI*, vol 3, no 6, pp. 653-660, Nov 1983.

[BAR78] H. Barrow and J. Tenenbaum, "Recovering intrinsic scene characteristics from images", *Computer Vision Systems*, A.R. Hanson and E.M. Riseman, eds., Academic Press, New York, pp. 3-35, 1978.

[BAR81] H. Barrow and J. Tenenbaum, "Computational vision," *Proc. IEEE*, vol 69, no 5, pp. 572-595, 1981.

[BES86] P.J. Besl and R.C. Jain, "Segmentation Through Symbolic Surface Description" *Proceedings CVPR86*, Miami, June 1986, pp. 77-85.

[BEV89] J.R. Beveridge, R.S. Weiss, and E.M. Riseman, forthcoming technical report.

[BIE85] I. Biederman, "Human image understanding: Recent research and a theory", *Computer Vision, Graphics, and Image Processing*, vol. 32, pp. 29–73, 1985.

[BLU73] H. Blum, "Biological shape and visual science", *J. of Theoretical Biology*, vol 38, pp. 205-287, 1973.

[BOL86] M. Boldt and R. Weiss, "Geometric Grouping of Lines", CVPR, pp. 489–495, 1986.

[BOLL82] R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The Local-Feature-Focus Method", *International Journal of Robotics Research*, vol. 1, no. 3, pp. 57–82, 1982.

[BOW88] J. Stewman and K. Bowyer, "Creating the perspective projection aspect graph of polyhedral objects", *IEEE Second Int. Conf on Computer Vision*, Tampa, FL, pp. 494–500, Dec. 1988.

[BRA] M. Brady, J. Ponce, A. Yuille, and H. Asada, "Describing Surfaces", *Proceedings of 2nd International Symposium on Robotics Research*, Hanafusa and Inoue (Eds.), MIT press, Cambridge, Ma.

[BRA84] M. Brady and H. Asada, "Smoothed local symmetries and their implementation", *The First International Symposium on Robotics Research*, M. Brady and R. Paul, eds., MIT Press, Cambridge, 1984.

[BRO81a] R. Brooks, *Model-Based Computer Vision*, UMI Research Press, Ann Arbor, MI, 1981.

[BRO81b] R. A. Brooks, "Symbolic reasaoning among 3D models and 2D images", *Artificial Intelligence*, vol. 17, pp. 285–348, August 1981.

[BUR86] J.B. Burns, A.R. Hanson and E.M. Riseman, "Extracting Straight Lines", IEEE Trans. Pattern. Anal. Mach. Intell., v.8, no.4, pp. 425–455, Jul. 1986.

[BUR87a] J. B. Burns and L. J. Kitchen, "An approach to recognition in 2D images of 3D objects from large model bases", Technical Report 87-85, COINS Dept, University of Massachusetts, Amherst, MA 01003, August 1987.

[BUR87b] Burns, J. B. and L. J. Kitchen, "Recognition in 2D images of 3D objects from large model bases using prediction hierarchies", Proc. IJCAI-10, 1987.

[BUR89] J. B. Burns, forthcoming Phd Dissertation, University of Mass., Computer and Information Science.

[BUR83] P.J. Burt, C. Yen, and X. Xu, "Multi-resolution flow-through motion analysis", *IEEE CVPR Conference Proceedings*, pp.246-252, June 1983.

[CAL85] J. Callahan and R. Weiss, "A Model for Describing Surface Shape", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, June 1985, pp. 240-245.

[CAN86] J. Canny, "A computational approach to edge detection", *PAMI*, vol 8, no 6, pp. 679-698, 1986.

[DAV75] L. Davis, "A survey of edge detection techniques", *Computer Graphics and Image Processing*, vol 4, pp. 248-270, 1975.

[DOL88] J. Dolan, "The Perceptual Organization of Image Curves", Technical Report, in preparation, COINS Dept, University of Massachusetts, Amherst, MA 01003, 1988.

[DOL89] J. Dolan, and R. Weiss, forthcoming technical report.

[DRA87] B. Draper, R. Collins, J. Brolio, J. Griffith, A. Hanson and E.M. Riseman, "Tools and Experiments in the Knowledge-Based Interpretation of Road Scenes", *Proceedings of the DARPA Image Understanding Workshop*, January 1987.

[DRA89] B. Draper, R. Collins, J. Brolio, J. Griffith, A. Hanson, and E. Riseman, "The Schema System", *International Journal on Computer Vision*, 2, pp. 209-250, 1989.

[DUD78] S. Dudani and A. Luk, "Locating straight-line edge segments on outdoor scenes." *Pattern Recognition*, vol 10, no 3, pp. 145-157, 1978.

[FER85] F.P. Ferrie, and M.D. Levine, "Piecing Together the 3D Shape of Moving Objects: An Overview", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, June 1985, pp. 574-584.

[FIS83] M. Fischler and R. Bolles, "Perceptual organization and the curve partitioning problem", *Proceedings IJCAI 1983*, pp.1014–1018.

[GIG88] Z. Gigus, J. Canny, R. Seidel, "Efficiently computing and representing aspect graphs of polyhedral objects", *IEEE Second Int. Conf on Computer Vision*, Tampa, FL, pp. 30–39, Dec. 1988.

[HAN87] A.R. Hanson and E.M. Riseman, "The VISIONS Image Understanding System", in *Advances in Computer Vision*, Chris Brown, Ed., Erlbaum Press, 1987.

[HAR84] R.M. Haralick, "Digital step edges from zero crossings of second directional derivatives", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol 6, pp. 58-68, 1984.

[HER84] M. Herman and T. Kanade, "The 3D MOSAIC scene understanding system: Incremental reconstruction of 3D scenes from complex images", *DARPA Image Understanding Workshop*, pp. 137-148, 1984.

[HOF83] D.D. Hoffman and W. Richards, "Parts of Recognition", MIT AI Memo 732, Dec. 1983.

[HON83] T.H. Hong, M.O. Shneier, R.L. Hartley, and A. Rosenfeld, "Using pyramids to detect good continuation", *IEEE Transactions on Systems, Man, and Cybernetics*, vol 13, pp. 631-635, 1983.

[HUT87] D.P. Huttenlocher and S. Ullman, "Object recognition using alignment", *IEEE Proc. First Int. Conf. on Computer Vision*, London, pp. 102-111, June 1987.

[KAM87] Behzad Kamgar-Parsi and Behrooz Kamgar-Parsi, "A Nonparametric Method for Fitting a Straight Line to a Noisy Image", University of Maryland Technical Report, CAR-TR-315, 1987.

[KAN77] T. Kanade, "Model representation and control structures in image understanding," *Proc IJCAI-5*, August 1977.

[KER81] Y.L. Kergosien, "La famille des projections orthogonales d'une surface et ses singularities", *C.R. Acad. Sc. Paris*, 292 , pp. 929–932, 1981.

[KOE76] J.J. Koenderink and van Doorn, A.J., "The singularities of the visual mapping", *Biological Cybernetics*, 24, pp. 51–59, 1976.

[KOE84] Jan J. Koenderink, "What Does the Occluding Contour Tell Us About Solid Shape?", *Perception*, vol 13, 1984, pp. 321-330.

[LEE78] E. Leeuwenberg, "Quantification of ertain visual pattern properties: Salience, transparency, similarity" *Formal Theories of Visual Perception*, E. Leeuwenberg and H. Buffart, eds., John Wiley and Sons, pp. 277-298, 1978.

[LOW82] D.G. Lowe, and T.O. Binford, "Segmentation and aggregation: An approach to figure ground phenomena" *DARPA Image Understanding Workshop*, 1982, pp. 168-178.

[LOW83] D.G. Lowe and T.O. Binford, "The perceptual organization as a basis for visual recognition", *Proceedings of AAAI-83*, Washington, DC, pp. 255-260, 1983.

[LOW85] D.G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.

[LOW87] D. G. Lowe, "The viewpoint consistency constraint", *International Journal of Computer Vision*, vol. 1, pp. 57-72, 1987.

[MAR82] D. Marr, *Vision*, Freeman, San Francisco, 1982.

[MCC80] C. McCrory, "Profiles of surfaces", preprint, University of Warwick, England, 1980.

[MCK85] D.M. McKeown and J.F. Pane, "Alignment and Connection of Fragmented Linear Features in Aerial Imagery", CMU Tech. Report CMU-CS-85-122, 1985.

[NEV80] R. Nevatia, and K.R. Babu, "Linear feature extraction and description", *CGIP*, v.13, pp. 257-269, 1980.

[ONE66] B. O'Neill, *Elementary Differential Geometry*, Academic Press, New York, 1966.

[REY87] G. Reynolds, and J.R. Beveridge, "Searching For Geometric Structure in Images of Natural Scenes", UMASS COINS Tech. Report 87-03, 1987.

[RIS86] E.M. Riseman and A.R. Hanson, "A Methodology for the Development of General Knowledge-Based Vision Systems", UMASS, COINS Tech. Report 86-27, July 1986.

[RIS87] E.M. Riseman, A.R. Hanson, and R. Belknap, "The Information Fusion Problem: Forming Token Aggregations Across Multiple Representations", UMASS COINS Tech. Report 87-44, 1987.

[ROS] R. Langevin, G. Levitt, and H. Rosenberg, "Herissons et Multiherissons", preprint.

[ROS86] A. Rosenfeld, "Pyramid algorithms for perceptual organization", *Behavior Research Methods, Instruments, and Computers*, vol 18, no 6, pp. 595-600, 1986.

[SCO84] R. Scott, "Graphics and prediction from models", *Proceedings: I U Workshop*, pp. 98-106, 1984.

[STA88] L. Stark, D. Eggert, and K. Bowyer, "Aspect graphs and nonlinear optimization in 3-D object recognition", *IEEE Second Int. Conf on Computer Vision*, Tampa, FL, pp. 501-507, Dec. 1988.

[STE87] K.A. Stevens, and A. Brookes, "Detecting structure by symbolic constructions on tokens", *Proc. CVGIP*, vol 37, pp. 238-260, 1987.

[THO87] D. Thompson and J. Mundy, "Three-dimensional model matching from an unconstrained viewpoint", *Proceedings IEEE International Conference on Robotics and Automation*, pp 208-220, 1987.

[TSO82] J. Tsotsos, "Knowledge of the visual process: Content, form and use," *Proc. 6th International Conference on Pattern Recognition*, pp. 654-669, Oct. 1982.

[TSO86] J. Tsotsos, "Representational axes and temporal cooperative computation," in M. Arbib and A. Hanson (eds.), *Vision, Brain and Cooperative Computation*, MIT Press, Cambridge, MA, 1986.

[WEI86] I. Weiss, "Straight Line Fitting in a Noisy Image", University of Maryland Technical Report, CAR-TR-234, 1986.

[WEIS86] R. Weiss and M. Boldt,"Geometric grouping applied to straight lines", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami Beach, pp. 489-495, June 1986.

[WEIS85] R. Weiss, E.M. Riseman, and A.R. Hanson,"Geometric grouping of straight lines", *Proc. DARPA Image Understanding Workshop*, Miami Beach, FL, pp. 443-447, 1985.

[WER23] M. Wertheimer, "Untersuchungen zur Lehrer von der Gestalt. II," *Psychologische Forschung*, vol 4, pp. 301-350, 1923.

[WIL88] L.R. Williams and A.R. Hanson, "Translating optical flow into token matches and depth from looming," *Proc. Int. Conf. on Computer Vision*, Tampa, pp. 441-448, April 1988.

[WIT83] A. Witkin "Scale-space filtering",*Proceedings of IJCAI*, pp. 1019-1021, Karlsruhe, 1983.

[YUI83] A.L. Yuille, and T. Poggio, "Scaling Theorems for Zero-crossings", MIT A.I. Memo 730, 1983.